Decentralized Multiagent Trajectory Planning

Cheng Chang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2025

Committee:

Robert Breidenthal

Mehran Mesbahi

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Decentralized Multiagent Trajectory Planning

Cheng Chang

Chairs of the Supervisory Committee:
Robert Breidenthal
Aeronautics and Astronautics

Mehran Mesbahi
Aeronautics and Astronautics

In this work, we present novel approaches to solve the decentralized trajectory planning for networked multiagent systems. We first provide the new reformulation of the safety constraint for faster convergence with guaranteed safety, with the iteration to converge being a fraction of traditional linearization approaches. This newly proposed constraint reformulation is based on the combination of the dual problem to the minimum distance between convex sets and biconvex optimization. The second important result obtained is the fully decentralized, scalable trajectory optimization algorithm based on the reformulated biconvex optimization, along with the successive convexification (SCVX) for handling the nonlinear dynamics. This algorithm features a minimum sharing data in comparison to the traditional distributed gradient method or dual decomposition-based algorithm like alternating direction method of multipliers (ADMM). The number of variables shared is less than half that of ADMM-SCP in our setups. We also showed the strong convergence of our biconvex method to the partial minimum under mild assumptions. Numerical results with up to ten agents and hardware experiments were performed to validate our claims. This work will enable efficient coordination of large-scale cooperating unmanned vehicles navigating in complex environments.

TABLE OF CONTENTS

Page

LIST OF FIGURES

ACKNOWLEDGMENTS

I want to thank the following factors that play important roles in my education path and life in general: my parents for providing me with an good initial condition, my friends in the RAIN lab for giving me the correct directions so I can get closer to the goal, Professor Mehran Mesbahi for building useful objectives for me to optimize, and Professor Robert Breidenthal for adding more perturbation(which is good!) so I can explore more possibilities outside the area of control. Without their support along this journey, I wouldn't be able to reach this somewhat good local optimal condition.

Chapter 1

INTRODUCTION

In the past two decades, the industry has witnessed a rapid development of small single-board computers (SBCs) in terms of computational speed and communication capability [**?** 2]. These affordable and scalable unmanned vehicles with advanced SBCs have been deployed in various environments for tasks such as surveillance, rescue, and logistic systems. Also, we have seen an ever-rising demand for fully autonomous driving vehicles operating in non/cooperative environments. However, one of the fundamental challenges in decentralized multiagent control is to ensure a collision-free path between agents and obstacles, in which agents are commanded to navigate through complex and dynamic environments in a cooperative fashion to achieve different tasks by sharing local information across agents in the network [3]. To avoid the collision, algorithms ranging from the simple Rapidly-exploring random trees with certain dynamics (RRT)[4] or artificial potential field (APF) [5], to the state-of-the-art sequential convex programming were developed.

To ensure the collision-free path, we will be focused on the convex optimization-based algorithm approach, which, when compared with other non-optimization approaches, can not only provide a "potentially" better solution with clear analytical convergence properties, but also can give a solution that, most of the time, satisfies the constraints imposed on the system. Lastly, due to the advancement of SBCs, individual agents can host fairly complex optimization solvers on board, with solving time less than a second[6, 7, 8, 9]. Most existing schemes to solve this problem are centralized, which makes the system relatively unreliable, as it requires a powerful computation center to solve a large problem, and the system will fail if such a center is disabled. Some parallel methods are therefore created to increase the robustness of the system and reduce the computation load on the center, which can be further categorized into weak centralized and decentralized methods [**?**]. Also, optimization-based algorithm allows us to include other objectives, such as formation and

different tasks, into our problem formulation. In this work, we will focus on the decentralized scheme for scalability and system robustness compared to other methods [**?**] . The focus of this work is to develop distributed partial primary partial dual optimization algorithms to generate collision-free paths in complex environments with a group of cooperative agents. We will begin with the literature review on safe trajectory planning commonly used on distributed systems and distributed optimization methods, which are the fundamentals for the distributed trajectory optimization problem.

1.1 Safe Trajectory Planning

To ensure such a non-convex constraint is respected for the safety of the vehicles, there are many proposed methods ranging from the simple source-sink like artificial potential field (APF) [5, 1] to the state of art sequential convex programming (SCP)[10, 6, 7, 8] methods. In this section, we will give an overview on the past efforts to enforce the safety constraints for multiagent systems.

1.1.1 Artificial Potential Field

Artificial potential field, as its name suggests, creates a field in the space by treating obstacles and the starting location as the sources which create repulsive forces while placing a sink in the space acting as the target position that attracts the agent. Therefore, the fields will be composed of several "peaks" trying to push the agents away while the target location will be the lowest part of this field, and the designed controller in the agent will follow this potential map to "descend" to some minimum or stationary points as given in Fig 1.1 Let $x_{des} \in \mathbb{R}^3$ be the target location, $x \in \mathbb{R}^3$ be the position of the agent, $v \in \mathbb{R}^3$ be the control velocity, and $x_{obs} \in \mathbb{R}^3, r \in \mathbb{R}^{++}$ be the position and the radius of the obstacle. The simplest realization of the APF method is given as:

$$v = \alpha(x_{des} - x) + \beta(r)(x - x_{obs}) \tag{1.1}$$

where $\alpha \in \mathbb{R}^{++}$ is the proportional gain for the velocity and $\beta(r) : \mathbb{R} \to \mathbb{R}$ is a value function to decide the strength of the repulsive velocity.

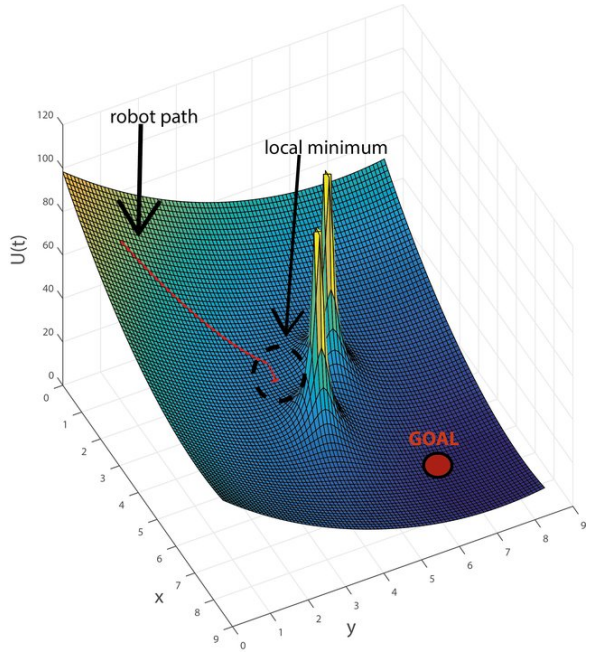

Figure 1.1: Illustration of APF method [1]

1.1.2 Control Barrier Function

Control barrier function method is also one of the most common approaches to ensure the agent's states stay within some invariant set [11, 12, 13]. In the context of obstacle avoidance, the safe set is usually defined as $S := \{X(t)|h(x(t)) \geq 0\}$, where $X(t), x_{obs} \in \mathbb{R}^3, h(X(t)) = \|X(t) - x_{obs}\| - d_{safe}$ denotes the positions of agent, obstacle in the space, and the value function associated measuring the distance between the agent and obstacle. The idea is that if $\nabla h(X(t))^T f(x(t), u(t)) \geq 0,\ \forall X(t) \in B := \{X|h(X(t)) = 0\}$, then any forward sequence of $X(t + dt), dt \geq 0$ will be in the set S with dynamic defined by $\dot{x(t)} = f(x(t), u(t))$, where $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m$. In most of the literature for this type of constraint, the safe constraint is written as:

$$\nabla h(X(t))^T f(x(t), u(t)) \geq -\alpha h(X(t)) \tag{1.2}$$

where $\alpha \geq 0$ is the parameter to be adjusted. In the case of the single integrator dynamics, this happens to be the quadratic constraint.

1.1.3 Summary of the safe trajectory planning methods

The table below illustrates the limitations and advantages of commonly used trajectory planning methods.

Table 1.1: Comparison of safe trajectory planning approaches

	Characteristics		
Method	Complexity	Constraints satisfaction	Predictable trajectory
Optimization			
CBF	Medium (real-time QP)	Yes	Only one step
SCP	High (solve complete optimization problem)	Yes	Yes
Others			
RRT	Low (random search)	No	Yes
APF	Low	No	No

1.1.4 Sequential Convex Programming

Many works from this department have focused on this type of approach by convexifying the constraints into different forms to be solved by the convex solver [10, 6, 7, 8], which approximates the constraints using affine functions. More details on this approach will be given in Chapter 3.

1.2 Multiagent Trajectory Planning and Decentralized Optimization

Currently, there are two approaches to address multiagent trajectory planning: centralized and decentralized pursuits. In this section, we will compare these two methods and discuss the challenges faced by the two strategies.

1.2.1 Centralized multiagent trajectory planning

The centralized strategy is usually solved in the same fashion as the single-agent case by lumping variables associated with all agents into one large optimization problem, which, in many cases, will have nice convergence properties and first-order optimality conditions as in the single-agent optimization case. These advantages make it suitable for problems with small numbers of agents; in contrast, this approach doesn't scale well as the number of variables and constraints will grow at least linearly with agent number N, and lastly, this approach requires one powerful computation node to send the solution to each individual agent, which makes the entire multiagent network let robust against system failure. The centralized scheme is illustrated below, where one powerful station will collect all the information and solve a single large optimization problem before passing the solutions to each agent.

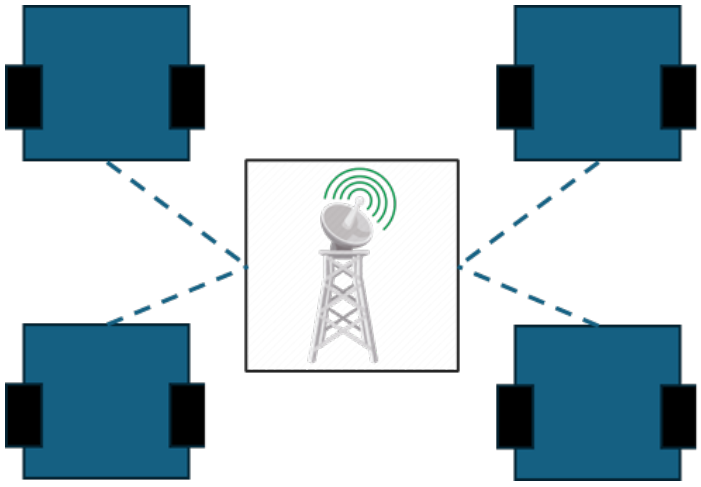

Figure 1.2: Illustration of centralized scheme

1.2.2 Decentralized multiagent trajectory planning

To overcome the drawbacks of the centralized method, many algorithms for decentralized planning have been developed. As seen in the figure, in the decentralized case, there is no "station" required to solve a single problem; instead, all agents participate in solving part

of the larger problem. This method allows the problems to be solved at smaller scales while increasing the fault tolerance of the entire system. One thing worth mentioning is that some parallel algorithms, like the original ADMM are actually not fully decentralized, as they still require part of the procedures to be solved in a centralized way, and in this example, it is usually the auxiliary variable minimization step.

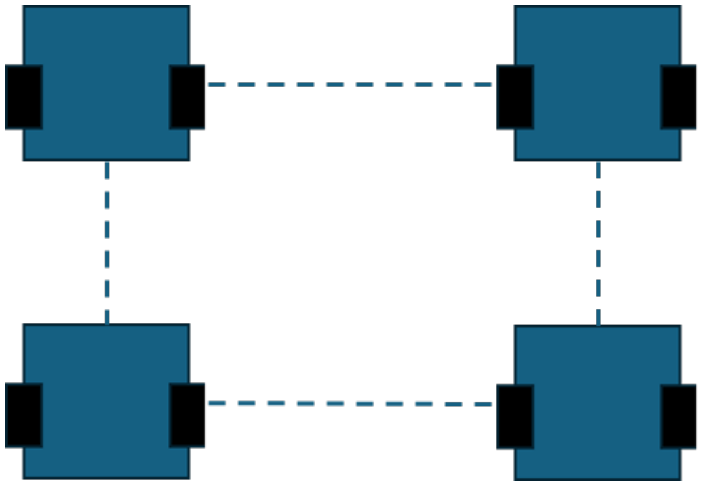

Figure 1.3: Illustration of decentralized scheme

1.2.3 Evolution of decentralized optimization

Several strategies for solving the decentralized optimization with non-convex constraints and cost functions have been proposed. Currently, there are two mainstream methods for solving general optimization problems in a decentralized fashion: **Primal** formulation approaches and **Dual** reformulation approaches.

We will give a brief introduction to the important discoveries on decentralized optimization for both primal and dual methods, and, of course, there are some methods that partially use primal and dual formulations to solve this problem, and the algorithm to be presented in this work belongs to such a category.

Primal approach

A substantial amount of work has been devoted to solve the distributed optimization

with primal descent approach, especially in the edge computing domain where the objective functions are usually nonconvex and servers will perform optimization over a time-varying network to find the optimal solution to the objective function. Gossip algorithm, in which the gradient descent and consensus are combined, usually involves performing an optimization step, followed by a consensus step, sequentially, in the hope that the decision variables will converge to an agreement and reach some local minima. Suppose we have a decomposable objective function with local objectives $f : \mathbb{R}^i \to \mathbb{R}$ only accessible to local agents as follows:

$$\min_x \sum_{i=1}^{N} f^i(x^i) \tag{1.3}$$

where $x^i \in \mathbb{R}^n$ is the variables associated with each agent, and the algorithm will perform following repeatably with a step size of $\alpha \in \mathbb{R}^+$ and "mixing coefficients" $a_{ij} \in \mathbb{R}, (i,j) \in V \times V, j \in \mathcal{N}_\rangle$:

$$\widetilde{x}_k^i = x_k^i - \alpha \nabla f^i(x_k^i) \tag{1.4a}$$

$$x_{k+1}^i = \sum_{l \in \mathcal{N}_i} a_{il} \widetilde{x_k^l} \tag{1.4b}$$

in which the first step will try to solve the optimal value for the local problem (can also be projected and stochastic gradient descent), followed by the second step to mix variables through the network. Common assumptions made on these types of algorithm are: strongly-connected graph, ℓ_1 smooth on the objective function, and diminishing step size. The main convergence results for variants of this come from the fact that as step size decreases, the consensus error defined by $e^{ij} = \|x^i - x^j\|, i \neq j$ will decrease monotonically. In the end, when $eij \ \leq \delta$, the local variables will converge to the variable centroid by $\|x_k^c - x_k^i\| \leq \delta$, where $x_k^c = \frac{\sum_{i=1}^N x_k^i}{N}$ if there are N agents with uniformly-weighted network graph. The resulting average dynamic then becomes:

$$F(x_{k+1}^c) \leq F(x_k^c) - \alpha \|\nabla F(x_k^c)\|^2 + \sum_{i=1}^{N} g(\|d_k^i\|) \tag{1.5}$$

where $g(\|d_k^i\|) : \mathbb{R} \to \mathbb{R}, d_k^i = x_k^c - x_k^i$ are the class $\mathcal{K}$ function and disagreeableness vectors. One can see that as the consensus error is small enough compared to the local gradient, the

sequence will decrease. For more detailed information, please refer to [14, 15, 16, 17]. In [14, 17], the convergence of distributed stochastic gradient descent (DSGD) on the non-convex objective with non-convex constraints is established by defining the projection function that projects the non-convex set into a convex subset. [16] showed that the convergence of DSGD doesn't depend on the diminishing step size; instead, it only requires that the perturbation (moment) be bounded with a constant step size. Both works represent the current view on the convergence of distributed optimization by using a two-step analysis: showing consensus in the primal variables across agents and then studying the evolution of the system centroid, which typically leads to the first-order optimality condition when the gradient across agents reaches an agreement.

[15] used a unique Lyapunov method to construct a positive value function and showed that the monotonic decrease of such a function occurs before reaching the neighborhood of local minima.

Dual approach

Instead of performing gradient descent on the primal problems, there are also many works approaching the problem from the dual, and conducting dual ascent on either the Lagrangian or augmented Lagrangian. For linear programming, one of the simplest distributed optimization methods is the dual decomposition method, in which one first performs the optimization on the primal variables in a distributed way, then performs consensus on all variables before performing the gradient ascent on the dual variables. Consider a simple LP:

$$\begin{aligned} &\min_{x} \sum_{i=1}^{N} f_i(x_k^i) \\ &\text{s.t.} Ax_k = b \end{aligned} \tag{1.6}$$

where $x = [x^1, \ldots, x^N]^T$, the dual decomposition method uses Lagrangian $L_i(x_k^i, y_k^i)$ associated with each x_k^i to perform the primal variable optimization. The algorithm takes the

form of:

$$\widetilde{x_k^i} = argmin_{x_k^i} L_i(x_k^i, y_k^i) \tag{1.7a}$$

$$x_{k+1}^i = \sum_{l \in \mathcal{N}_i} a_{il} \widetilde{x_k^l}, \widetilde{y_k^i} = \sum_{l \in \mathcal{N}_i} a_{il} \widetilde{y_k^l} \tag{1.7b}$$

$$y_{k+1}^i = \widetilde{y_k^i} + \alpha \nabla L_i(x_{k+1}^i, \widetilde{y_k^i}) \tag{1.7c}$$

the second step in this algorithm acts similarly to that of the distributed gradient descent to mix the variables with neighbors to bring down the consensus error. Details are given in [9, 18, 19]. As this method requires many additional assumptions and converges slowly, another distributed method called the Alternating Direction Method of Multipliers or ADMM was developed by using the auxiliary variables to reformulate the constraints with augmented Lagrangian for the applications on the non-differentiable objective functions with faster convergence, and the general form of ADMM with separable constraints and convex objective functions has the syntax:

$$\begin{aligned} &\min_x \sum_i^N f_i(x_k^i) \\ &\text{s.t. } \sum_i^N A^i x_k^i = b \\ &h_i(x_k^i) \leq 0, \forall i \end{aligned} \tag{1.8}$$

The constraints are usually handled by the indicator function

$$I(r) = \begin{cases} 0 & \text{if } r = 0 \\ \infty & \text{otherwise} \end{cases}$$

, and the objective function will be replaced with an extended objective function

$$g_i(y^i) = \begin{cases} f_i(y^i) \leq 0 & \text{if } h_i(y^i) \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

. The augmented Lagrangian is then given by:

$$\begin{aligned} &L_\rho(x, y, \lambda) = \\ &I(\sum_i^N A^i x_k^i - b) + \sum_i^N (g_i(y_k^i) + (\lambda_k^i)^T A^i (y_k^i - x_k^i) + \frac{\rho}{2} \|A^i (y_k^i - x_k^i)\|^2) \end{aligned} \tag{1.9}$$

, where y is the auxiliary variable associated with constraints, while λ is the multiplier for the equality constraints. The algorithm will then follow a similar approach to multiplier decomposition by solving the following in the order: solving the y-minimization, performing λ maximization for the dual ascent, and solving for the x-minimization to complete one iteration. See [20] for more details.

Readers can refer to [9, 21, 22, 23, 20] for more information. Although ADMM is only applicable to the convex objective function, SQP methods are frequently used in conjunction with ADMM and the consensus algorithm to handle the non-convexity of the objective function and constraints with a super-linear convergence rate. One of the most famous examples is the Augmented Lagrangian-based Alternating Direction Inexact Newton Method (ALADIN method), which enables solving distributed non-convex optimization problems with continuous, twice-differentiable objective functions. See algorithm 2 in [23] for the detailed procedure.

Since ALADIN requires the nonlinear solvers for the first step of the algorithm, [22] essentially replaces the first step with the SQP method so as to achieve a faster convergence rate. In [22], the authors combine the SQP with ADMM to perform the distributed trajectory optimization and showed that the solution will converge to the stationary point at a super-linear rate by using the convergence properties of ADMM.

For the majority of the dual approach to solve the non-convex problems, SQP is required to solve the minimization over the primal and auxiliary variables. Additionally, the requirement for twice-differentiability in most ALADIN-based algorithms also limits their applications.

1.2.4 Comparison of distributed optimization methods

In the previous discussion on the existing methods for solving the distributed optimization, all the methods require a complete estimate of all variables, which in the context of trajectory optimization will require each agent to store and share N copies of local information, which results in poor scalability when the trajectory time steps are large with complex dynamics. The convergence guarantee for most existing distributed optimization

algorithms relies on agreement to the first order, which means the gradients of either the primary objective function or the Lagrangian should reach consensus for the convergence to the stationary point. Our algorithm, although it lacks such a clean convergence property, requires less data to be communicated between agents, which makes it scale better as they only need the agreement on the estimated positions' history of other agents instead of the whole optimization variables.

Below shows a comparison of the convergence properties and data requirements for different algorithms.

Table 1.2: Comparison of centralized and decentralized methods

Method	Modeling conditions		
	Objective function / Lagrangian	Data shared	Convergence
Centralized			
SCP [24, 6, 8, 9]	Lipschitz	Full variables	Stationary point
Decentralized			
Primal method [17, 25, 14, 15]	Lipschitz	Full variables	Stationary point
Dual method [23, 26, 22]	Twice differentiable	Primal variables + Auxiliary variable + Dual variables+Twice differentiable	Stationary point
This work	Lipschitz	Partial states	Uncertain in general/ Partial minimum with no consensus error

1.3 Contribution of this work

In this work, our main contributions are: 1. provide a review of the theoretical foundation on various topics relevant to the distributed trajectory planning; 2. present a novel bi-level dual approach for reformulating the safe constraint with convergence guarantee; 3. demonstrate a new scalable distributed trajectory planning method with a simple convergence analysis; lastly we verify our claims with both numerical experiments up to 10 agents forming complex

geometry and hardware experiments with nonlinear unicycle dynamics that represents a large class of ground robots.

1.4 Notation convention in this work

This section introduces some of commonly used notations across the work, and more notations specific to certain chapters will be defined later. We denote the number of agents by $N \in \mathbb{Z}$, and the underlying graph associated with the network topology of agents by $\mathcal{G} = (V, E)$, where $V = \{1, 2, 3, \ldots, N\}$ be the vertices or agent index, and $E \in V \times V$ be the edge set of the graph, and $\mathcal{N}_i$ represents the neighbor set of agent i. $\{x_k^i(t), u_k^i(t)\}$ are the states and control input for agent i at time $t \in [0, T]$ for iteration $k \in \mathbb{Z}$. We use the superscript to denote the agent index and subscripts are referred to the iteration index. There are several variants for the iteration index, and reader should refer to the descriptions in later chapters for their definitions.

We now define the important variables representing the concatenations of different quantities. Let

$$\mathbf{x}_k^i = \begin{bmatrix} x_k^i(0) & \ldots & x_k^i(T) \end{bmatrix}^\intercal \in \mathbb{R}^{n(T+1)} \tag{1.10}$$

be the concatenation states for agent i. Also, we define the estimate of states of agent j by agent i by:

$$\mathbf{x}_k^{ij} = \begin{bmatrix} x_k^{ij}(0) & \ldots & x_k^{ij}(T) \end{bmatrix}^\intercal \in \mathbb{R}^{n(T+1)} \tag{1.11}$$

and we will set $x_k^{ii} := x_k^i$, as agent i possesses the true state of its own.

Same as the states variables, we also have the concatenated control input as:

$$\mathbf{u}_k^i = \begin{bmatrix} u_k^i(0) & \ldots & u_k^i(T-1) \end{bmatrix}^\intercal \in \mathbb{R}^{nT} \tag{1.12}$$

Let the concatenation of estimations **from** agent i at iteration k be denoted

$$\mathbf{X}_k^i := \begin{bmatrix} \mathbf{x}_k^{i1} \\ \ldots \\ \mathbf{x}_k^{iN} \end{bmatrix} \in \mathbb{R}^{N \times n(T+1)}. \tag{1.13}$$

This quantity X_k^i can be thought of as the total information that agent i has of the network. This induces the following local optimal control problem We also need another concatenated

variable to store all the true states of all agents in the network

$$\mathbf{X}_k := \begin{bmatrix} \mathbf{x}_k^1 \\ \dots \\ \mathbf{x}_k^N \end{bmatrix} \in \mathbb{R}^{N \times n(T+1)}. \tag{1.14}$$

.

For the compact representation of consensus dynamic, we let Y_k^i be concatenation of prediction from all agents on the agent i:

$$Y_k^i := \begin{bmatrix} \mathbf{x}_k^{1i} \\ \dots \\ \mathbf{x}_k^{Ni} \end{bmatrix} \in \mathbb{R}^{N \times n(T+1)}. \tag{1.15}$$

We also define the consensus error resulting from the agent i's prediction of the agent j's state trajectory as $\mathbf{e}_k^{ij} = \|\mathbf{x}_k^{ij} - \mathbf{x}_k^j\|$. The global consensus error on estimation of agent i's trajectory is now obtained as:

$$\mathbf{e}_k^i := \| \left[\mathbf{e}_k^{1i}, \dots, \mathbf{e}_k^{Ni} \right] \| \in \mathbb{R}. \tag{1.16}$$

Lastly, when we use norm without subscript, it usually represents the ℓ_2 norm.

1.5 Thesis Structure

In Chapter 2, we give the explanation on our problem formulation, which contains two scenarios: the single agent trajectory planning and multiagent trajectory planning

In Chapter 3, we will give short introductions to the theoretical foundation we will use throughout the work, which includes two main topics: graph theory and convex optimization, and relevant theorems to be used.

In Chapter 4, we will revisit the problem statements and present our main algorithm built upon theories to be introduced in Chapter 3.

In Chapter 5, convergence proofs will be given for both the our original SVM reformulation method in single agent case and multiagent trajectory optimization.

Finally in Chapters 6 and 7, we will present our results and comparisons between our methods against standard SCvx method.

Chapter 2

PROBLEM STATEMENTS

This chapter will give the formal setups of our problems regarding the trajectory planning for single agent and decentralized multiagent settings.

2.1 Single agent trajectory optimization

Following we formulate the singe agent trajectory planning problem with general nonlinear discrete system dynamic $x(t+1) = f(x(t), u(t))$ and obstacle avoidance constraints, where $r, R \in \mathbb{R}^{++}$ are the radius of agent and obstacle j, $x^j(t) \in \mathbb{R}^2$ is the position of obstacle j at time t, and $x(t, 0:2)$ indicates the position of agent at t, as we assign the first two elements of states be the coordinates in $\mathbb{R}^2$ space. The cost $C(\mathbf{x}, \mathbf{u})$ will be the sum of the running cost and the terminal cost in the form of $C(\mathbf{u}) = \|\mathbf{u}\| + \phi(x(T))$, where $\phi(x) : \mathbb{R}^n \to \mathbb{R}$ is the convex terminal cost.

$$
\begin{aligned}
&\min_{(\mathbf{x},\mathbf{u})} C(\mathbf{x}, \mathbf{u}) \\
&\text{s.t. } x(t+1) = f(x(t), u(t)), \\
&\qquad \forall t = 0, \ldots, T-1 \\
&\qquad r + R - \|x(t, 0:2) - x^j(t)\| \leq 0, \forall t = 0, \ldots, T \\
&\qquad x(0) = \bar{x}(0) \\
&\qquad x(T) = \bar{x}(T)
\end{aligned}
\tag{2.1}
$$

2.2 Multiagent decentralized trajectory optimization

Similar to the single agent case, and we assume that the all agents have uniform radius and dynamics; we can extend our single agent problem simply by multiplying the cost function and constraints by N times, and we want to minimize the sum of individual cost. To simplify

the notation, we denote $s^{ij}(x^i(t), x^j(t)) = 2r - \|x^i(t, 0:2) - x^j(t, 0:2)\|$.

$$\begin{aligned}
&\min_{(\mathbf{x}^i, \mathbf{u}^i)} \sum_{i=1}^{N} C(\mathbf{x}^i, \mathbf{u}^i) \\
&\text{s.t. } x^i(t+1) = f(x^i(t), u^i(t)), \\
&\quad \forall t = 0, \ldots, T-1 \\
&\quad s^{ij}(x^i(t), x^j(t)) \leq 0, , \forall t = 0, \ldots, T \\
&\quad x^i(0) = \bar{x}^i(0) \\
&\quad x^i(T) = \bar{x}^i(T)
\end{aligned} \tag{2.2}$$

where $(i, j) \in V$ represent the agent index. Note that in the multiagent setting, $x^j(t)$ is no longer the position of obstacle, it now represents the states of agent j at time t. (2.2) It is clearly the centralized problem. To solve this problem in a decentralized fashion, we separate this problem into N local problems to be solved by each agent independently at iteration k.

$$\begin{aligned}
&\min_{(\mathbf{x}^i_k, \mathbf{u}^i_k)} C(\mathbf{x}^i_k, \mathbf{u}^i_k) \\
&\text{s.t. } x^i_k(t+1) = f(x^i_k(t), u^i_k(t)), \\
&\quad \forall t = 0, \ldots, T-1 \\
&\quad s^{ij}(x^i_k(t), x^{ij}_k(t)) \leq 0, \forall t = 0, \ldots, T \\
&\quad x^i_k(0) = \bar{x}^i_0(0) \\
&\quad x^i_k(T) = \bar{x}^i_0(T) \\
&\quad \|\mathbf{x}^i_k - \mathbf{x}^i_{k-1}\| \leq r^i_k
\end{aligned} \tag{2.3}$$

where r^i_k is the consensus trust region to regulate the consensus error among agents, which can be determined by some mechanisms to be discussed later, and k is the algorithmic iteration index to be introduced later. From now on, we will use $s^{ij}(t) = s^{ij}(x^i_k(t), x^{ij}_k(t))$ to simplify the expression. To further simplify the problem, we assume our agents have uniform, continuous, linear time variant dynamic represented as follows:

$$\begin{aligned}
&\dot{x} = A^i(t)x(t) + B^i(t)u(t) \\
&A^i(t) = \frac{\partial f(x^i, u^i)}{\partial x^i}|_{x^i(t), u^i(t)}, B^i(t) = \frac{\partial f(x^i, u^i)}{\partial x^i}|_{x^i(t), u^i(t)}
\end{aligned} \tag{2.4}$$

The discrete system can therefore be formulated as:

$$\begin{aligned} x^i(t+1) &= A_d^i(t)x^i(t) + B_d^i(t)u^i(t) \\ A_d^i &= \exp(A^i(t)dt), B_d^i = \int_0^{dt} \exp(A^i(\tau)\tau)d\tau B \end{aligned} \tag{2.5}$$

With the linear dynamic, we observe that the only non-convex constraint in these optimization problems is the collision-free constrains. We also introduce the penalized version of this local problem by defining the nonlinear penalty function associated with the each agent

$$\begin{aligned} &J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) = C(\mathbf{x}_k^i, \mathbf{u}_k^i) + \\ &\sum_{t=0}^{T-1} [\lambda \|x_k^i(t+1) - f(x_k^i(t), u_k^i(t))\|_1 + \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t))] \end{aligned} \tag{2.6}$$

Although the last term $\sum_{j=1,j\neq i}^{N} \lambda \max(0, s_k^{ij}(t))$ is zero for local problem if the solution is feasible, it is an important measurement of cost related to the inaccurate estimate of other agent's trajectories. With the linearized dynamics and the Lagrangian cost function, the actual local problem that agent will solve is given as

$$\begin{aligned} \min_{(\mathbf{x}_k^i, \mathbf{u}_k^i)} \; & J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) \\ \text{s.t. } & x^i(t+1) = A_d^i(t)x^i(t) + B_d^i(t)u^i(t), \\ & \forall t = 0, \ldots, T-1 \\ & s^{ij}(t) \leq 0, \forall t = 0, \ldots, T \\ & x_k^i(0) = \bar{x}_0^i(0) \\ & x_k^i(T) = \bar{x}_0^i(T) \\ & \|\mathbf{x}_k^i - \mathbf{x}_{k-1}^i\| \leq r_k^i \end{aligned} \tag{2.7}$$

which are subject to further reformulation as one can notice that the problem is still non-convex, and the quality of linearized dynamics also depend on the trust region γ we will impose later on the subproblems of this local problem. In Chapter (3), we will further reformulate above problems into proper convex subproblems with SVM reformulations.

Chapter 3

THEORETICAL FOUNDATIONS

Two main themes will be focused in this chapter: Graph theory and Convex optimization.

3.1 Graph theory

Graph theory is widely used across different sciences and engineering disciplines, from the everyday power-grid network design problem to the optimal origami design and control. Graph, commonly used as a representation of networked systems, where the linking parts are indicated by the edge sets $E \in V \times V$, and the nodes that participate in the decision making are referred to as vertices $V^i, \in 1, \ldots, N$, contains many important properties of the given networked system. Depending on the direction of interaction for a pairs of nodes, we have undirected graph and directed graph. In this work, we assumed the nodes (agents) can always achieve the two way communication, which can be conveniently represented by the undirected graph. Readers are strongly recommended to read [3, 24] for more applications on the graph theories and different versions of consensus protocols.

3.1.1 Continuous and discrete consensus protocol

The static consensus protocol for a continuous, networked system composed of N agents is given as:

$$\dot{x}^i(t) = -\sum_{j \in \mathcal{N}(\rangle)} (x^i(t) - x^j(t)) \tag{3.1}$$

If one expand the system with state vector $\mathbf{x} = [x^1, \ldots, x^N]^T$ and work out the system dynamic, one will have the LTI system in the form of $\dot{\mathbf{x}} = -L\mathbf{x}$, where $L \in S_+^N$ is the Laplacian representing the network dynamics. By the construction of Laplacian, it has the following properties: 1. $\lambda_{min}(L) = 0, \lambda_i(L) = 0$ iff $i = 0$, with ascertaining indexing order for eigen values; 2. the eigenvector v_0 associated with the minimum eigenvector is $\frac{\vec{1}}{\sqrt{N}}$.

With these properties, the system will converge exponentially to the average of initial state $\frac{\vec{1}^T\mathbf{x}(t=0)\vec{1}}{N}$. The complete proof can be found in [3].

The discrete version follows the same fashion, and the dynamic of which is given by

$$\mathbf{x}(t+1) = (I - \epsilon L)\mathbf{x}(t) \tag{3.2}$$

where $\epsilon \leq \frac{1}{N}$ is required for the Perron matrix $P = I - \epsilon L \in S_+^n$ to have $\lambda_{max} \leq 1$. Also, this dynamic will drive the system to $\frac{\vec{1}^T\mathbf{x}(t=0)\vec{1}}{N}$ in the linear rate.

3.1.2 Modified consensus protocol

We will introduce a slightly modified consensus protocol, which converge to, instead of the average of the initial condition, the specific value of the agent, as we desire the agent's estimated trajectories will converge the true trajectories possessed only by the local agents. We set

$$\mathbf{x}_{k+1}^{ij} = \mathbf{x}_k^{ij} - \epsilon \sum_{l \in \mathcal{N}_i} (\mathbf{x}_k^{ij} - \mathbf{x}_k^{lj}) \tag{3.3}$$

for $i \neq j$. Note that $\epsilon \leq \frac{1}{N}$ for the stability of consensus dynamic. And of course, we will automatically set $\mathbf{x}_{k+1}^{ii} := \mathbf{x}_{k+1}^{i}$.

Our goal is to write the above consensus dynamic into a series of compact linear-time-invariant systems. To do so, we have to deinfe modifying matrices:

$$\mathbb{A}1^i = [e^1, \ldots, e^{i-1}, \vec{0}, e^{i+1}, \ldots, e^N] \tag{3.4a}$$

$$\mathbb{A}2^i = [0, \ldots, 0, e^i, 0, \ldots, 0]^T \tag{3.4b}$$

$$\mathbb{P}^i = \mathbb{A}1^i P + \mathbb{A}2^i \tag{3.4c}$$

where e^i is the coordinate vector with i-th element being 1 and the rest being 0, and the modified Perron matrix $\mathbb{P}^i \in \mathbb{R}^{N \times N}$ for estimation of agent i's trajectory.

To give a concrete example on the structure of modification matrices, let's consider a distributed system composed of four agents with a connected graph, the matrices $\mathbb{A}1^2, \mathbb{A}2^2$

and modified Perron matrix $\mathbb{P}^3$ for agent 3 are now defined as:

$$\mathbb{A}1^3 = diag([1,1,0,1]) \tag{3.5a}$$

$$\mathbb{A}2^3 = diag([0,0,1,0]) \tag{3.5b}$$

$$\mathbb{P}^3 = \mathbb{A}1^3 P + \mathbb{A}2^3 \tag{3.5c}$$

The modification matrices $\mathbb{A}1, \mathbb{A}2$ will slightly modify the consensus dynamic by ensuring each agent will retain its own copy of trajectory and let direct neighbors access the true trajectory information.

During each consensus iteration, there will be N linear systems representing the consensus dynamics for the estimation of trajectories for each agent as follows:

$$\begin{aligned} Y_{k+1}^1 &= \mathbb{P}^1 Y_k^1 \\ &\cdots\cdots \\ Y_{k+1}^N &= \mathbb{P}^N Y_k^N \end{aligned} \tag{3.6}$$

The definition of Y_k^i is given by (1.15).

3.2 Trajectory optimization

3.2.1 Convex optimization

Disciplined Convex Programming(DCP), refers to a class of optimization revolving around solving such problems with both 1. its objective being a convex (concave) function, and 2. its constraints being convex (concave), is one of the most powerful optimization techniques developed, as solving any such problem will guarantee to converge to global minimum set to be defined [9].

The most common form of convex optimization is given as follows:

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } Ax + B = 0 \\ &\qquad h(x) \leq 0 \end{aligned} \tag{3.7}$$

where $f(x), h(x,y)$ are both convex functions, with the affine equality constraints. With this DCP, according to [9], the solution to this problem will converge to the global minimum

sets $S_{opt} := \{x^* | Ax^* + b = 0, h(x^*) \leq 0, f(x^*) \leq f(x),\ \forall x \in \mathcal{F}\}$, where $\mathcal{F} := \{x | Ax + B = 0, h(x) \leq 0\}$.

There are many subclasses of programming associated with DCP, such as linear programming(LP), quadratic programming(QP), second-order cone programming(SOCP), and semi-definite programming (SDP). Throughout this work, we will be using SOCP extensively as it is one of the more general programming types than LP and QP.

3.2.2 Alternating Convex Search (ACS)

ACS method, along with bilevel optimization method, has been of interest for solving various bilinear and biconvex problems when the original optimization problems are difficult to solve with coupling variables [27, 28, 18], but can be simplified to convex subproblems if some of the variables are fixed. This idea of repeatedly fixing some variables while optimizing others is the central spirit of ACS.

Consider the following example biconvex problem:

$$\begin{aligned} &\min_{x,y} f(x,y) \\ &\text{s.t. } axy + b = 0 \\ &\quad h(x,y) \leq 0 \\ &\quad x \in X \subseteq \mathbb{R}, y \in Y \subseteq \mathbb{R} \end{aligned} \tag{3.8}$$

We assume that sets X and Y are both nonempty and compact, and $h(x,y)$ is convex when either x or y is fixed. Lastly, we assume $h(x,y)$ is a continuous function. We first define the x,y-sections of the jointly feasible set $\mathcal{F} := \{(x,y) | axy + b = 0, h(x,y) \leq 0\}$, denoted by

$$\begin{aligned} B_x &= \{\bar{y} \in Y, (x,\bar{y}) \in \mathcal{F}\} \\ B_y &= \{\bar{x} \in X, (\bar{x},y) \in \mathcal{F}\} \end{aligned} \tag{3.9}$$

We then define the partial minimum of the objective function by:

$$\begin{aligned} f(x^*,y^*) &\leq f(x,y^*),\ \forall x \in B_{y^*} \\ f(x^*,y^*) &\leq f(x^*,y),\ \forall y \in B_{x^*} \end{aligned} \tag{3.10}$$

Finally, we have the theorem stating the strong convergence of a sequence of solutions to the partial minimum with the above assumptions. Also, one should notice that a stationary point within feasible set $\mathcal{F}$ must be a partial minimum if $f(x, y)$ is biconvex.

Proposition: partial minimum is a stationary point Suppose $f(x, y)$ is continuously differentiable, a partial minium point (x^*, y^*) is also a stationary point to f.

Proof: Because f is biconvex, we have

$$\begin{aligned} f(x, y^*) &\geq f(x^*, y^*) + \nabla_x f(x, y^*)^T (x - x^*) \\ f(x^*, y) &\geq f(x^*, y^*) + \nabla_y f(x^*, y)^T (y - y^*) \end{aligned} \tag{3.11}$$

at optimal, it is required that $\nabla_x f = \nabla_y f = 0$, we have $f(x, y^*) = f(x^*, y^*), f(x, y^*) = f(x^*, y^*)$ being the stationary point.

Theorem: Convergence to partial minimum

Consider the sequence $(x_k, y_k)|_{k=0}^{\infty}$, if the following holds

$$\begin{aligned} f(x_{k+1}, y_k) \leq f(x_k, y_k),\ \forall (x_k, y_k), (x_{k+1}, y_k) \in \mathcal{F} \\ f(x_{k+1}, y_{k+1}) \leq f(x_{k+1}, y_k),\ \forall (x_{k+1}, y_k), (x_{k+1}, y_{k+1}) \in \mathcal{F} \end{aligned} \tag{3.12}$$

then this sequence will converge to the partial minimum set defined by (5.2). For the detailed proof, please refer to [27], and the idea of the proof is to show the sequence is lower bounded by partial minimum while the sequence is decreasing monotonically, which leads to the strong convergence to such point. The final algorithm representing problems in the form of (3.8) is given in the following algorithm:

3.2.3 SVM Reformulation

One of the more common approachs to the non-convexity in the optimization is to reformulate those constraints in a more solvable way, for instance, convexifying through linearization or reformulating the constraints with its dual when the constraints themselves are also self-contained optimization problems. The SVM method, originally proposed in this work, as its name suggests, is inspired by the support vector machine used in data classification problems. In the SVM method, we reformulate the non-convex collision avoidance problem into a set of classification problems. Firstly, consider two agents

Algorithm 1 Alternative Convex Searching Method

procedure INITIALIZATION(x_0, y_0)

Generate initial guess(x_0, y_0)

end procedure

procedure MAIN LOOP(x_k, y_k)

while $\|[x_{k+1}, y_{k+1}] - [x_k, y_k]\| \geq Tol$ **do**

fix y_k and solve (3.8) to get x^*

$x_{k+1} \leftarrow x^*$

fix x_{k+1} and solve (3.8) to get y^*

$y_{k+1} \leftarrow y^*$

end while

end procedure

case where they have fixed shapes that can be represented as closed convex sets such that $x^i(t) \in S_1 := \{x^i(t) | f^l(x^i(t)) \leq 0, \forall l\}, x^j(t) \in S_2 := \{x^j(t) | g^l(x^j(t)) \leq 0, \forall l\}, l = 1, 2, \ldots, p$, and $f^l(x), g^l(x), l = 1, 2, \ldots, p$ are convex functions. We assume only agent i is performing the optimization and we fix agent j's trajectory. The obstacle avoidance constraints require that the following holds:

$$\begin{aligned} \inf_{(x^i(t), x^j(t))} & \|x^i(t) - x^j(t)\|_2 \geq d_{min} \\ \text{s.t. } & f^l(x^i(t)) \leq 0,\ l = 1, 2, \ldots, p \\ & g^l(x^j(t)) \leq 0,\ l = 1, 2, \ldots, p \end{aligned} \tag{3.13}$$

where d_{min} is the minimum safety distance between agents. One can then recognize that this is equivalent to finding $x^i(t)$ such that there exists a supporting slab with a minimum thickness d_{min} separating two convex sets. This concept can be visualized by viewing the agents as the convex data sets, and the goal is to find the support hyper "slab" with the minimum thickness of d_{min} to classify the agents into different categories in the space.

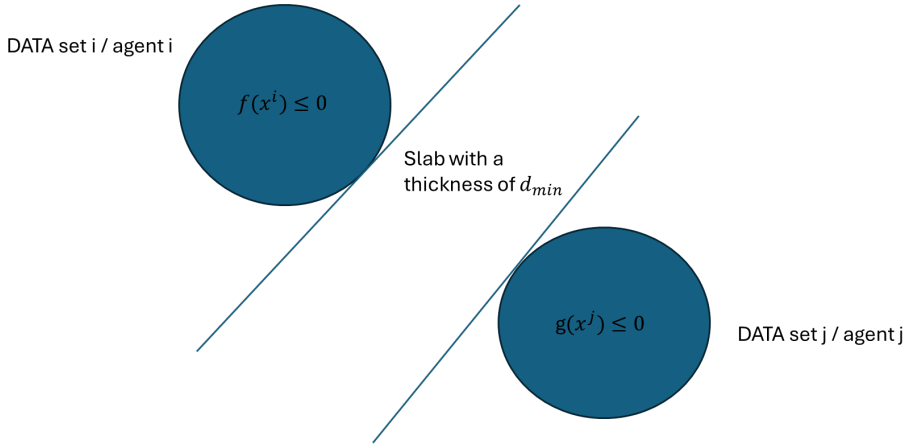

Figure 3.1: Visualization of SVM reformulation in $\mathbb{R}^2$ with two agents

Now, let's consider the dual of these constraints, which have the form:

$$\begin{aligned} &\sup_{z^{ij}(t)} \inf_{x^i(t)} (\sum_{l=1}^{p} \lambda_l f_l(x^i(t)) + z^T x^i(t)) + \\ &\qquad \inf_{x^j(t)} (\sum_{l=1}^{p} \mu_l g_l(x^j(t)) - z^T x^j(t)) \geq d_{min} \\ &\text{s.t. } \|z\|_2 \leq 1 \\ &\qquad \lambda \succeq 0, \mu \succeq 0 \end{aligned} \tag{3.14}$$

where $z, \lambda, \mu \in \mathbb{R}^2$ are Lagrangian multipliers associated with equality and inequality constraints. Now we consider the shape of agents to be circular with radius r^i, r^j and denote the centroids of the circles: $x_k^i(t, 0:2), x_k^j(t, 0:2)$ as the trajectories of both agents, we have $f = \|A^i x^i(t) + b^i(t)\|_2 - 1, g = \|A^j x^j(t) + b^j(t)\|_2 - 1, b_k^i(t) = -A^i x_k^i(t, 0:2) r^i, b_k^j(t) = -A^j x_k^j(t, 0:2) r^j$, where $A^i, A^j \in S_{++}^n$ are matrices defining the shapes of the agents. After lengthy derivation (derivation is given in Appendix A) and assuming $r^i = r$, the dual problem to the constraints becomes:

$$\begin{aligned} &\sup_{z_k^{ij}(t)} z_k^{ij}(t)^T (b^j(t) - b^i(t)) r - 2r \|z_k^{ij}(t)\|_2 \geq d_{min} \\ &\text{s.t. } \|z_k^{ij}(t)\|_2 \leq 1 \end{aligned} \tag{3.15}$$

We can reformulate the constraints $\sup_{z_k^{ij}(t)} z_k^{ij}(t)^T (b^j(t) - b^i(t)) - 2 * (r + \frac{d_{min}}{2}) \|z_k^{ij}(t)\|_2^2 \geq 0$

with $\|z_k^{ij}(t)\|_2 \leq 1$. Which seems trivial reformulation, but we will show how it becomes critical in making this problem into biconvex optimization problem in which both levels are convex in their own. We can reformulate (2.7) with the dual constraints described above as:

$$
\begin{aligned}
&\min_{(\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{z}_k^i)} C(\mathbf{x}_k^i, \mathbf{u}_k^i) \\
&\quad \text{s.t. } x_k^i(t+1) = A_k^i(t)x_k^i(t) + B_k^i(t)u_k^i(t), \\
&\qquad \forall t = 0, \ldots, T-1 \\
&\qquad x_k^i(0) = \bar{x}_0^i(0) \\
&\qquad x_k^i(T) = \bar{x}_0^i(T) \\
&\quad \sup_{z_k^{ij}(t)} z_k^{ij}(t)^T(bj(t) + A^i x_k^i(t, 0:2)) - 2(r + \frac{d_{min}}{2})\|z_k^{ij}(t)\|_2^2 \geq 0, \\
&\qquad \forall t = 0, \ldots, T \\
&\qquad \text{s.t. } \|z_k^{ij}(t)\|_2 \leq 1, \forall t = 0, \ldots, T
\end{aligned}
\tag{3.16}
$$

One can see that $x_k^i(t)$ is the linking variable that appears in both the upper level and lower level problems, which makes this bilevel problem still non-convex. At this step, it almost becomes feasibility checking problem in that if there exist Lagrangian multiplier $z_k^{ij}(t)$ such that $Q(z_k^{ij}(t), x_k^i(t)) = z_k^{ij}(t)^T(bj(t) - bi(t)) - 2(r + \frac{d_{min}}{2})\|z_k^{ij}(t)\| \geq 0$, then the collision avoidance is guaranteed. Note that we can remove the lower level problem as $Q(z_k^{ij}(t), x_k^i(t)) \geq 0$ will automatically guarantee $\sup_{z_k^{ij}(t)} Q(z_k^{ij}(t), x_k^i(t)) \geq 0$, we can

further simplify the problem with discrete linear dynamics as:

$$\begin{aligned}
&\min_{(\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{z}_k^i)} C(\mathbf{x}_k^i, \mathbf{u}_k^i) \\
&\quad \text{s.t. } x_k^i(t+1) = A_k^i(t)x_k^i(t) + B_k^i(t)u_k^i(t), \\
&\qquad \forall t = 0, \ldots, T-1, \\
&\qquad x_k^i(0) = \bar{x}_0^i(0) \\
&\qquad x_k^i(T) = \bar{x}_0^i(T) \\
&\qquad Q(z_k^{ij}(t), x_k^i(t)) \geq 0, \forall t = 0, \ldots, T \\
&\qquad \|z_k^{ij}(t)\|_2 \leq 1, \forall t = 0, \ldots, T
\end{aligned} \tag{3.17}$$

where $\mathbf{z}_k^i \in \mathbb{R}^{N \times 2 \times T}$ is the concatenation of all Lagrangian multipliers associated with agent i, and A_k^i, B_k^i are the discrete representation of nonlinear continuous dynamics given by (2.5). One also should notice that the only part that makes this problem non-convex is the term $z_k^{ij}(t)^T bi(t) = -z_k^{ij}(t)^T A^i x_k^i(t)$, which is bilinear if we separate this bilevel problem into two independent problems. We also notice that previously, $x^i(t, 0:2) \in X_{old}^i := \{x^i(t, 0:2) | 2r - \|x^j(t, 0:2)\| \leq 0\}$ is not convex for any choice of $x^j(t, 0:2)$; now, with the reformulation, we have $x^i(t, 0:2) \in X_{SVM}^i := \{x^i(t, 0:2) | z_k^{ij}(t)^T (bj(t) + A^i x^i(t, 0:2)) - 2(r + \frac{d_{min}}{2})\|z_k^{ij}(t)\|_2^2 \geq 0\}$, which results in a convex set for any given $z_k^{ij}(t)$. Therefore, we transformed our nonconvex constraints with a nonconvex feasible set for $x(t)$ into a bilinear constraint with a biconvex feasible set. So the feasible set of the reformulated problem also becomes biconvex. To address this issue, we add the slack variables $s_k^{ij} = -Q(z_k^{ij})$ and augment our objective by formulating a nonlinear penalty function $J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i)$ as our new objective. From (5.1), we notice the coupling relationship between $s_k^{ij}(t)$ and $x_k^i(t)$, which now leads us to the alternative direction method (ACS) as this problem can be decoupled into two convex subproblems. The first problem will perform primal variables minimization

$x(t), u(t)$ with $\mathbf{z}_k^i$ treated as constants, and serves as the leader in this bilevel optimization.

$$\begin{aligned}
\min_{(\mathbf{x}_k^i, \mathbf{u}_k^i)} \; & C(\mathbf{x}_k^i, \mathbf{u}_k^i) + \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t)) \\
\text{s.t. } & x_k^i(t+1) = A_k^i(t) x_k^i(t) + B_k^i(t) u_k^i(t), \\
& \forall t = 0, \ldots, T-1, \\
& x_k^i(0) = \bar{x}_0^i(0) \\
& x_k^i(T) = \bar{x}_0^i(T) \\
& s_k^{ij}(t) = -Q(z^{ij}(t), x_k^i(t)), \forall t = 0, \ldots, T \\
& s_k^{ij}(t) \leq 0, \forall t = 0, \ldots, T
\end{aligned} \tag{3.18}$$

The second optimization problem is updating the Lagrange multipliers, which finds valid Lagrange multipliers to ensure the collision avoidance constraints are satisfied. With the same fashion as in the first problem, we treat the state variables $x(t), u(t)$ as constants.

$$\begin{aligned}
\min_{(\mathbf{z}_k^i)} \; & \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t)) \\
\text{s.t. } & s_k^{ij}(t) = -Q(z_k^{ij}(t), x_k^i(t)), \\
& \forall t = 0, \ldots, T-1, \\
& \|z_k^{ij}(t)\|_2 \leq 1, \forall t = 0, \ldots, T
\end{aligned} \tag{3.19}$$

where

$$\mathbf{z}_k^i = \begin{bmatrix}
z_k^{i1}(0), & \ldots, & z_k^{i1}(0), & \ldots, & z_k^{i1}(T), & \ldots, & z_k^{i1}(T) \\
\ldots, & \ldots, & \ldots, & \ldots, & \ldots, & \ldots, & \ldots, \\
z_k^{iN}(0), & \ldots, & z_k^{iN}(0), & \ldots, & z_k^{iN}(T), & \ldots, & z_k^{iN}(T)
\end{bmatrix} \in \mathbb{R}^{N \times (N(T+1))} \tag{3.20}$$

is the stacked Lagrangian multipliers for agent i. The $\max(0, s(t))$ is used to update the multipliers that violate the safety constraints. Note that if we don't consider the optimality and drop the max operator, we can use the analytical solution for analytical solution $z^*(t) = \frac{b(t)}{\|b\|}, b(t) = bj(t) - bi(t)$ and gives the optimal value of $-\lambda(\|b(t)\|_2 - (2r + d_{min}))$, which will result in a larger "barrier" between agents, and this is especially useful with

limited computational power and want to achieve a robust safety trajectory. Also, for some nonlinear dynamics, one may need to add the trust region constraints to the algorithm commonly used to ensure the quality of the linear representation to the nonlinear system. To the end, we present our two-step SVM method for collision avoidance trajectory planning. Please do not confuse the iteration index k in the above algorithm with other k used in the

Algorithm 2 SVM reformulation for single agent

procedure INITIALIZATION($\mathbf{x}_0^i, \mathbf{u}_0^i$)

Generate initial states (can be infeasible) $\mathbf{x}_0^i, \mathbf{u}_0^i$

end procedure

procedure MAIN LOOP($\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{z}_k^i$)

while $\|\mathbf{x}_{k+1}^i - \mathbf{x}_k^i\| \geq Tol$ **do**

Solve (3.18) to get $\mathbf{x}_k^*, \mathbf{u}_k^*$

$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_k^*, \mathbf{u}_{k+1}^i \leftarrow \mathbf{u}_k^*$

Solve (3.19) with $\mathbf{x}_{k+1}^i, \mathbf{u}_{k+1}^i$ to get $\mathbf{z}_k^*$

$\mathbf{z}_{k+1}^i \leftarrow \mathbf{z}_k^*$

end while

end procedure

multiagent algorithm. Since the SVM reformulation only takes care of the safe constraints and the system is nonlinear, general SCP procedures are still needed to solve the local problems. In the chapter 6, we will show that by using this reformulation, a faster convergence rate can be achieved compared to the naive constraint linearization. Below, we will give a short introduction to the SCVX used in this work.

3.2.4 Successive Convexification (SCvx)

Before introducing the SCvx, we will construct the nonlinear penalty function and its corresponding problem. Let

$$\begin{aligned} &J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) = C(\mathbf{x}_k^i, \mathbf{u}_k^i) + \\ &\sum_{t=0}^{T-1} [\lambda \| x_k^i(t+1) - f(x_k^i(t), u_k^i(t)) \|_1 + \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t))] \end{aligned} \tag{3.21}$$

be the nonlinear penalty function, we have the penalized optimization problem as:

$$\min_{(\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{d}_k^i)} J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) \tag{3.22}$$

, which is an unconstrained optimization problem with non-convex objective.

SCVX is a trust region-based first-order approach to solving the universal nonlinear and non-convex optimization problem, and it gives the form of solving the linearized problems sequentially as follows:

$$\begin{aligned} &\min_{(\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i)} L(\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i) \\ &\text{s.t. } \|\mathbf{w}_k^i\| \leq \gamma_k^i \end{aligned} \tag{3.23}$$

where $L(\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i) : \mathbb{R}^{n\times(T+1)} \times \mathbb{R}^{m\times T} \times \mathbb{R}^{N\times 2\times(T+1)} \times \mathbb{R}^{n\times(T+1)} \to \mathbb{R}$ is the linear penalty function to be defined , $\mathbf{s}_k'^{ij}(t) = [s_k^{ij}(0), \ldots, s_k^{ij}(T)] \in \mathbb{R}^{T+1},\ \forall i, j \in V$, and the $\mathbf{d}_k^i \in \mathbb{R}^n, \mathbf{w}_k^i \in \mathbb{R}^m$ are the optimal increments of states and control over each optimization iteration, $\mathbf{v}_k^i = [v_k^i(0) \ldots, v_k^i(T)] \in \mathbb{R}^{n\times(T+1)},\ v_k^i(t) = x_k^i(t+1) + d_k^i(t+1) - f(x_k^i(t), u_k^i(t)) - A_k^i(t)d_k^i(t) - B_k^i(t)w_k^i(t) \in \mathbb{R}^n$ are virtual control terms for accounting dynamic infeasibility resulting from the linearization of nonlinear dynamics, where $A_k^i(t), B_k^i(t)$ are the discrete matrix representations of the nonlinear, continuous system, as can be seen from (2.5). Note that. We also denote $s_k'^{ij}(t) = s_k^{ij}(t) + \frac{\partial s_k^{ij}(t)}{\partial x^i(t)}|_{x_k^i(t)} d_k^i(t)$ as the slack variables accounting for the nonconvex constraints (in this case it is the collision avoidance constraints). The linear penalty function resembles the nonlinear version and is given by:

$$\begin{aligned} &L(\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i) = C(\mathbf{x}_k^i + \mathbf{d}_k^i, \mathbf{u}_k^i + \mathbf{w}_k^i) + \\ &\sum_{t=0}^{T-1} \lambda \|v_k^i(t)\|_1 + \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t)) \end{aligned} \tag{3.24}$$

In SCvx, we use the linearized and discretized system to represent the continuous and nonlinear system dynamics; we also linearize all the non-convex constraints to make them become the affine inequality constraints. The trust region mentioned in this subsection is to regulate the "instability" incurred by the linearization as a consequence of approximating nonlinear functions with first-order representations based upon the previous solution to (3.22). For more information on SCvx and its application, please refer to [6, 7, 8, 10]. Below we represent a simple SCvx with modified trust region regulation logic.

Algorithm 3 SCvx with simplified trust region updating method

procedure INITIALIZATION($\mathbf{x}_0^i, \mathbf{u}_0^i$)

Generate initial states (can be infeasible) $\mathbf{x}_0^i, \mathbf{u}_0^i$

end procedure

procedure MAIN LOOP($\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i$)

while $\|\mathbf{x}_{k+1}^i - \mathbf{x}_k^i\| \geq Tol$ **do**

Solve (3.23) to get $\mathbf{d}_k^i, \mathbf{w}_k^i$

if $L(\mathbf{d}_k^i, \mathbf{w}_k^i, \mathbf{s}_k'^{ij}, \mathbf{v}_k^i) \geq L(\mathbf{d}_{k+1}^i, \mathbf{w}_{k+1}^i, \mathbf{s}_{k+1}'^{ij}, \mathbf{v}_{k+1}^i)$ **then**

$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_k^i + \mathbf{d}_k^i$

$\mathbf{u}_{k+1}^i \leftarrow \mathbf{u}_k^i + \mathbf{w}_k^i$

$\gamma_{k+1}^i \leftarrow \gamma_k^i$

else

$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_k^i$

$\mathbf{u}_{k+1}^i \leftarrow \mathbf{u}_k^i$

$\gamma_{k+1}^i \leftarrow \frac{\gamma_k^i}{\omega_{SCvx}}$

end if

$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_k^*, \mathbf{u}_{k+1}^i \leftarrow \mathbf{u}_k^*$

end while

end procedure

SCVX will be used to solve the upper-level problem (primal minimization) in our bilevel optimization problem given in (3.18,3.19), while the lower-level convex subproblem (dual

update) will be solved directly.

Both SCvx and SVM reformulation are just sub-routines for single-agent trajectory planning, assisting in solving the multi-agent algorithm to be presented. Note that all the iteration index k used in this subsection will be relabeled as k_{opt} in the next section to prevent confusion.

Chapter 4
MAIN ALGORITHM

This chapter will be dedicated to the workflow of our main algorithm for the decentralized trajectory planning, which will combine the single agent optimization algorithms mentioned in previous chapters with the modified consensus protocol.

4.1 Decentralized multiagent trajectory planning

In the pursuit of decentralization, each agent will perform the followings for each algorithmic iteration $k \geq 0, k \in \mathbb{Z}$:1. evaluate the penalty cost associated with the consensus error; 2. solve the local optimization problems (2.7) for $k_{opt} \geq 0, k_{opt} \in \mathbb{Z}$ times until local problems converges, and the local problems can also be decomposed into more subproblems such as Algorithms[2,3]; 3. perform modified consensus algorithm for $k_{con} \geq 0, k_{con} \in \mathbb{Z}$ times to update the estimates of trajectories for other agents. Also, we denote the optimal solution to (2.7) is $\mathbf{x}_k^{i*} = \mathbf{x}_k^i + \mathbf{d}_k^i, \mathbf{u}_k^{i*} = \mathbf{u}_k^i + \mathbf{w}_k^i$, and $r_k^i \geq 0$ is the consensus trust region for the states and first mentioned in (2.3). To regulate the consensus trust region for the convergence, we need $r_{k+1}^i = \frac{r_k^i}{\omega}, \omega \geq 0$ when the nonlinear penalty increase due to consensus error is detected by the agent with local estimate. The main algorithm is given on the next page, and the accompanying figure shows the workflow of the initialization and main loop. In the main algorithm, we represent the consensus loops with blue, optimization with cyan, and trust region update with red. Each subroutines come with its own convergence tolerance, such as $Tol_{main} \geq 0$ for the main loop,$Tol_{opt} \geq 0$ for the local optimization loop ,and $Tol_{con} \geq 0$ for the consensus error. Also, a figure illustrating the algorithm system level will be given.

Algorithm 4 Decentralized Trajectory Optimization

procedure INITIALIZATION($\mathbf{x}_0^i, \mathbf{X}_0^i, \mathbf{u}_0^i, r_0^i$)

Initialize the states estimation $\mathbf{x}_0^{ij} \leftarrow \vec{0} \in \mathbb{R}^{n(T+1)}$

Generate initial states (best if trajectories are non-touching) $\mathbf{x}_0^i$

while $\|\mathbf{X}_{k_{con}+1} - \mathbf{X}_{k_{con}}\| \geq Tol_{con}$ **do**

if $\|\mathbf{X}_{k_{con}+1} - \mathbf{X}_{k_{con}}\| \leq Tol_{con}$ **then**

Stop, and **return** $\mathbf{X}_{k_{con}}^i$

end if

perform (3.3) to get $\mathbf{X}_{k_{con}+1}^i$

$\mathbf{X}_{k_{con}}^i \leftarrow \mathbf{X}_{k_{con}+1}^i, k_{con} \leftarrow k_{con} + 1$

end while

$\mathbf{X}_1^i \leftarrow \mathbf{X}_{k_{con}}^i$

$k \leftarrow k + 1$

end procedure

procedure DIST-OPT($\mathbf{x}_k^i, \mathbf{X}_{k+1}, \mathbf{u}_k^i, r_k^i$)

while $\|\mathbf{x}_{k+1}^i - \mathbf{x}_k^i\| \geq Tol_{main}$ or safety constraints violation ($\|\mathbf{s}_k^i\| > 0$) **do**

if $\|\mathbf{x}_{k+1}^i - \mathbf{x}_k^i\| \leq Tol_{main}$ and ($\|\mathbf{s}_k^i\| \leq 0$) **then**

Stop and **return** $\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i$

end if

while $\|\mathbf{x}_{k_{opt}+1}^i - \mathbf{x}_{k_{opt}}^i\| \geq Tol_{opt}$ **do**

if $\|\mathbf{x}_{k_{opt}+1}^i - \mathbf{x}_{k_{opt}}^i\| \leq Tol_{opt}$ **then**

Stop, and **return** $\mathbf{x}_{k_{opt}}^i, \mathbf{u}_{k_{opt}}^i$

else

Solve (2.7) by using Algorithms [2,3] to get $(\mathbf{x}_{k_{opt}+1}^i, \mathbf{u}_{k_{opt}+1}^i)$

$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_{k_{opt}}^i, \mathbf{u}_{k+1}^i \leftarrow \mathbf{u}_{k_{opt}}^i, k_{opt} \leftarrow k_{opt} + 1$

end if

end while

$\mathbf{x}^{i*} \leftarrow \mathbf{x}_{k_{opt}}^i, \mathbf{u}^{i*} \leftarrow \mathbf{u}_{k_{opt}}^i$

$d_k^i \leftarrow \mathbf{x}^{i*} - \mathbf{x}_k^i, w_k^i \leftarrow \mathbf{u}^{i*} - \mathbf{u}_k^i$

$\widetilde{\mathbf{x}_k^i} \leftarrow \mathbf{x}_k^i + \mathbf{d}_k^i, \|\mathbf{d}_k^i\| \leq r_k^i$

$\widetilde{\mathbf{u}_k^i} \leftarrow \mathbf{u}_k^i + \mathbf{w}_k^i$

$\widetilde{\mathbf{x}_k^{ij}} \leftarrow \mathbf{x}_k^{ij} + \vec{0}, \forall i \neq j$

while $\|\mathbf{X}_{k_{con}+1} - \mathbf{X}_{k_{con}}\| \geq Tol_{con}$ **do**

if $\|\mathbf{X}_{k_{con}+1} - \mathbf{X}_{k_{con}}\| \leq Tol_{con}$ **then**

Stop, and **return** $\mathbf{X}^i_{k_{con}}$

end if

perform (3.3) to get $\mathbf{X}^i_{k_{con}+1}$

$\mathbf{X}^i_{k_{con}} \leftarrow \mathbf{X}^i_{k_{con}+1}, k_{con} \leftarrow k_{con} + 1$

end while

$\mathbf{X}^i_{k+1} \leftarrow \mathbf{X}^i_k$

if $\frac{J(\widetilde{\mathbf{x}^i_k}, \mathbf{X}^i_{k+1}, \widetilde{\mathbf{u}^i_k})}{J(\widetilde{\mathbf{x}^i_k}, \mathbf{X}^i_k, \widetilde{\mathbf{u}^i_k})} \leq 1$ **then**

$r^i_{k+1} \leftarrow r^i_k$

$\mathbf{x}^i_{k+1} \leftarrow \widetilde{\mathbf{x}^i_k}$

$\mathbf{u}^i_{k+1} \leftarrow \widetilde{\mathbf{u}^i_k}$

else

if $r^i_k \geq r_{min}$ **then**

$r^l_{k+1} = \frac{r^l_k}{\omega}, \ \forall l \in i, \mathcal{N}_i$

else if $k \geq K_{flag}$ **then**

$r^i_{k+1} \leftarrow r_{min}$

else

$r^i_{k+1} \leftarrow r^i_k$

end if

end if

$k \leftarrow k + 1$

end while

end procedure

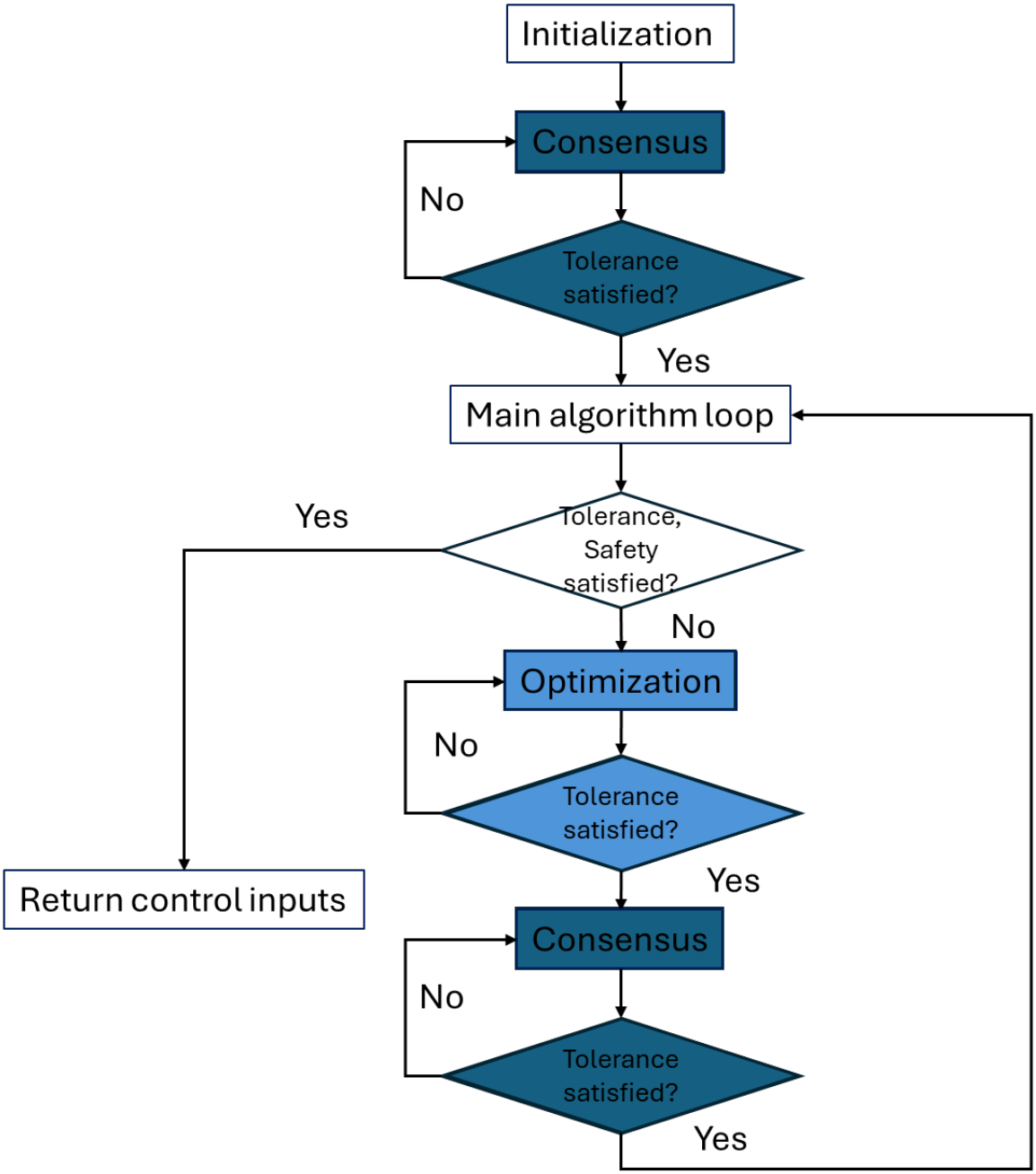

Figure 4.1: Diagram showing the workflow of the algorithm proposed in Algorithm 4

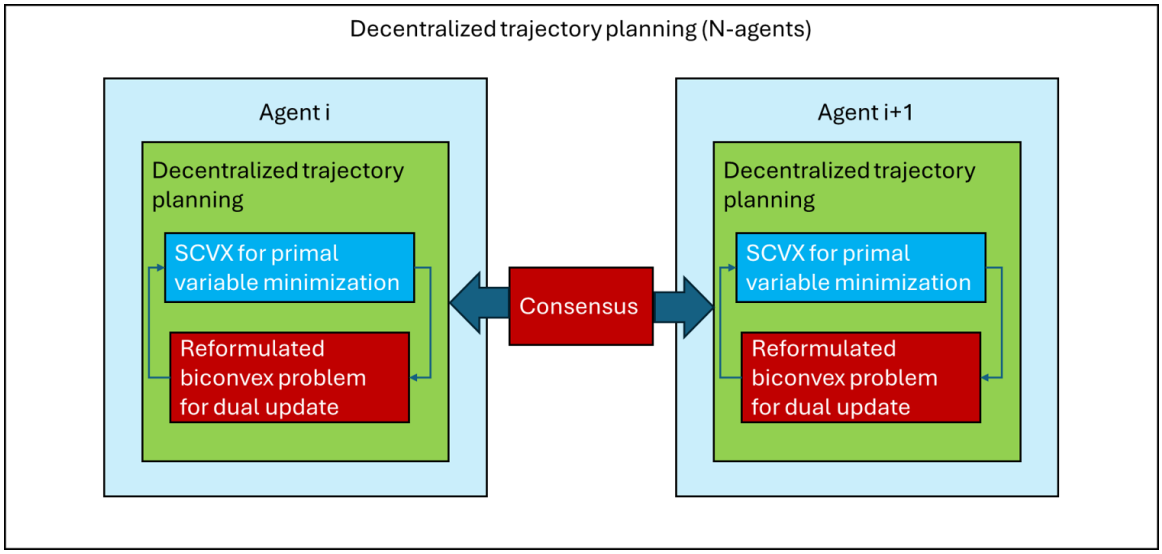

Figure 4.2: Algorithm level of decentralized trajectory planning

Chapter 5

CONVERGENCE ANALYSIS

We will show the strong convergence for both the single agent case and extend it to the multiagent scenario with additional assumptions on the problem setup.

5.1 Convergence of single agent trajectory planning

In this section, we discuss convergence proof of single agent trajectory planning algorithms with SVM reformulation will be given. For the convergence analysis of SCvx, reader can refer to [6, 7, 8, 10] for detailed proofs.

5.1.1 Convergence of the Alternating Convex Search approach

Recall that our local agent optimization problem is given as:

$$\begin{aligned}
\min_{(\mathbf{x}_k^i, \mathbf{u}_k^i, \mathbf{z}_k^i)} \; & C(\mathbf{x}_k^i, \mathbf{u}_k^i) + \sum_{j=1, j\neq i}^{N} \lambda \max(0, s_k^{ij}(t)) \\
\text{s.t. } & x_k^i(t+1) = A_k^i(t) x_k^i(t) + B_k^i(t) u_k^i(t), \\
& \forall t = 0, \ldots, T-1, \\
& x_k^i(0) = \bar{x}_0^i(0) \\
& x_k^i(T) = \bar{x}_0^i(T) \\
& Q(z_k^{ij}(t), x_k^i(t)) \geq 0, \forall t = 0, \ldots, T \\
& \|z_k^{ij}(t)\|_2 \leq 1, \forall t = 0, \ldots, T
\end{aligned} \tag{5.1}$$

which can be decomposed into bilevel optimization problems given by (3.18,3.19). We will establish the convergence results of this method for the LTI system by utilizing known convergence proofs for the ACS method. Also, to simplify the notation usage, we will use the single term $g(\mathbf{x}_k, \mathbf{u}_k) = \vec{0}$, where $g : \mathbb{R}^{nT} \times \mathbb{R}^{m(T-1)} \to \mathbb{R}^n$ to represent all affine constraints in our local problem.

Assumption 5.1.1: Non-empty and compact solutions sets:

We let $\mathcal{F} := \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) | g(\mathbf{x}_k, \mathbf{u}_k) = \vec{0}, Q(z_k^{ij}(t), x_k^i(t)) \geq 0, \|z_k^{ij}(t)\| \leq 1)\} \notin \emptyset$ be the compact jointly biconvex feasible set for the states, controls, and Lagrangian multipliers, respectively. Note that this set is in general not jointly convex in $(\mathbf{x}, \mathbf{u}, \mathbf{z})$.

Assumption 5.1.2: Linear independence constraint qualification:

We require that $\nabla g_x(\mathbf{x}_k, \mathbf{u}_k), \nabla g_u(\mathbf{x}_k, \mathbf{u}_k), \nabla Q_{active,z}, \nabla Q_{active,x}$ to be all linearly independent in columns.

Strong convergence: Under assumption 5.1.1, and 5.1.2. Let $[\mathbf{x}_k, \mathbf{u}_k]^T = \mathbf{y}_k \in \mathbb{R}^{(n+m)(T-1)+n}$ be the stacked upper level problem variables. The sequence $(\mathbf{y}_k, \mathbf{z}_k)|_{k=0}^{\infty}$ will strongly converge to the partial minimum defined by:

$$\begin{aligned} f(\mathbf{y}^*, \mathbf{z}^*) \leq f(\mathbf{y}, \mathbf{z}^*), \ \forall \mathbf{z} \in B_{\mathbf{y}^*} \\ f(\mathbf{y}^*, \mathbf{z}^*) \leq f(\mathbf{y}^*, \mathbf{z}), \ \forall \mathbf{y} \in B_{\mathbf{z}^*} \end{aligned} \tag{5.2}$$

where $B_{\mathbf{y}^*}, B_{\mathbf{z}^*}$ are the y,z-sections evaluated at the partial minimums of $\mathbf{y}^*, \mathbf{z}^*$.

Proof: The proof is the direct result of (3.2.2).

5.2 Convergence of multiagent trajectory planning

To simplify the analysis, we assume that the solutions given by the local solvers always have the properties described in the following assumptions; therefore, we essentially separate the analysis of the multiagent problem from the local optimization problems. Also, we only consider case where only one consensus operation is performed in each main iteration for the most of our analysis. Like the main algorithm, our proof can be divided into two parts: consensus and a noised-local optimization problem. First, we will show that the consensus error is bounded with some assumption on the solution to the nonlinear penalty function. The idea is that the estimation on each agent's states will converge to the truth plus some consensus error determined by the choice of consensus trust region.

Assumption 5.2.1: Existence of a bounded, sufficient decreasing solution to the Lipschitz conditioned nonlinear penalty function

We assume there always exists a solution $\mathbf{x}_k^{i*}, \mathbf{u}_k^{i*}$ to (3.22) such that the followings hold:

$$\|\mathbf{x}_k^{i*} - \mathbf{x}_k^i\| = \|\mathbf{d}_k^i\| \leq r_k^i \tag{5.3a}$$

$$J(\mathbf{x}_k^{i*}, \mathbf{X}_k^i, \mathbf{u}_k^{i*}) - J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) \leq -\beta\|\mathbf{d}_k^i\| \tag{5.3b}$$

Furthermore, by denoting $C = (\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i)^T$ as the concatenation of all data that agent i has, we assume $J(C_k^i)$ to be globally Lipschitz such that the following hold:

$$\|J(C_{k+1}^i) - J(C_k^i)\| \leq L_{max}\|C_{k+1}^i - C_k^i\| \tag{5.4a}$$

where L_{max} is the maximum Lipschitz constant associated with $J(C_k^i)$, $\forall k$.

Both assumptions are valid for some standard solving techniques with proper construction of cost functions, including SCvx; when some further assumptions are made, see [6, 7, 8] for more details. Furthermore, in our analysis, we assume no linearization error caused by the first-order approximation of SCVX or similar gradient-based algorithms.

Bounded consensus error:

After sufficient iterations, there exists K such that when $k \geq K$, the global consensus error for the estimation of agent i's states $\mathbf{e}_k^i \leq \delta$ for some $\delta \geq 0$, if trust region $r_k^i \leq \delta(\frac{1}{\lambda_2} - 1)$, where λ_2 is the second largest eigenvalue of $\mathbb{P}^i$.

Proof: Consider the standard Laplacian for a strongly connected graph with N agents $\in S^N$, which by the nature of construction has net zero in both row and column entries. Therefore, the resulting modified Perron matrix for discrete consensus has the form $\mathbb{P}^i = \mathbb{A}1^i(I - \epsilon) + \mathbb{A}2^i, \epsilon \leq 1/N$, which is a non-negative matrix with a spectrum of

$$\lambda_N \leq \lambda_{N-1} \leq \ldots \leq \lambda_2 \leq \lambda_1 = 1 \tag{5.5}$$

. Also, by construction, we have $\vec{1} \in (\mathbb{P}^i)$. One can notice that the modified Perron matrix replaces the original unit eigenvalue with the new unit eigenvalue introduced at $\mathbb{P}_{i,i}^i$. We can also diagonalize the modified Perron as $\mathbb{P}^i = Q^i D^i (Q^i)^{-1}$, where $Q^i, D^i \in \mathbb{R}^{N \times N}$ are the eigenbases and diagonal matrix encompassing all eigenvalues of $\mathbb{P}^i$, respectively.

Consider a static single value reference tracking scenario with modified consensus where all agents are tracking the time-invariant value of agent i:

$$y_{k+1} = Q^i (D^i)^{kcon} (Q^i)^{-1} y_0 \tag{5.6}$$

where $y_{kcon+1} = [y^1_{kcon} \dots y^i_{kcon} \dots y^N_{kcon}]^T \in \mathbb{R}^N$. We want to show that the modified consensus will converge at linear rate determined by λ_2 to a stationary truth y^i_{kcon}. We can rewrite the initial condition $C^i_0 = (Q^i)^{-1} y_0$ as the linear combination of eigenbases as $C^i_0 = \sum_{l=1}^N \alpha_l q_l$, where $\alpha \in \mathbb{R}, q \in \mathbb{R}^N$ are the coefficient and eigenbasis. Since all engenvalues except the one associated with eigenvector $\vec{1}$ reside within the unit circle, if we expand (5.6) as:

$$y_{kcon+1} = Q^i (D^i)^{kcon} [\alpha_1 q_1 + \dots + \alpha_i q_i + \dots + \alpha_N q_N] \tag{5.7}$$

it is easy to show that all the components will decrease to $\vec{0} \in \mathbb{R}^N$ linearly except $\alpha_i q_i = \alpha_i \vec{1} \in \mathbb{R}^N$, and the coefficient $\alpha_i = y^i$, which implies $\lim_{kcon \to \infty} y_{kcon} = y^i \vec{1}$ with consensus error decrease no slower than the linear rate of λ_2 by $\mathbf{e}^i_{kcon} \leq \lambda_2 \mathbf{e}^i_{kcon}$. This statements can easily be extended to general $Y^i_{kcon+1} \in \mathbb{R}^{N \times n(T+1)}$.

Assume after $k \geq K$ iterations, $\mathbf{e}^i_k \leq \delta$. According to our algorithm, the consensus error on agent i will decrease first by the consensus process, then increase by no more than $\|\mathbf{d}^i_k\| \leq r^i_k$ during the optimization process, the overall propagation of error in each step ($\widetilde{}$ denotes the intermediate step for optimization):

$$\widetilde{\mathbf{e}^i_k} \leq \mathbf{e}^i_k + \|\mathbf{d}^i_k\|, \ \|\mathbf{d}^i_k\| \leq r^i_k, \ \mathbf{e}^i_k \leq \delta \tag{5.8a}$$

$$\mathbf{e}^i_{k+1} \leq \lambda_2 \widetilde{\mathbf{e}^i_k} \tag{5.8b}$$

From the above, for the any choice of $\delta > 0$ such that $\mathbf{e}^i_{k+1} \leq \delta$, it is required that $\|\mathbf{d}^i_k\| \leq r^i_k \leq \delta(\frac{1}{\lambda_2} - 1)$, which completes the proof.

Bounded error for predicted nonlinear penalty function:

Under Assumption 5.2.1, if the global consensus error on agent i: $\mathbf{e}^i_k \leq \delta$, and the state inequality constraints are continuously differentiable, there exists a finite $L_{max} \geq 0$, such that the error

$$J_e = |J(\mathbf{x}^i_k, \mathbf{X}^i_k, \mathbf{u}^i_k) - J(\mathbf{x}^i_k, \mathbf{X}_k, \mathbf{u}^i_k)| \leq L_{max} \sqrt{N-1} \delta \tag{5.9}$$

Proof: Recall that (3.21) is comprised of local trajectory cost term: $C(\mathbf{x}^i_k, \mathbf{u}^i_k)$, dynamic violation term:$(\|x^i_k(t) - f(x^i_k(t), u^i_k(t))\|_1)$, and the state inequality constraints violation (collision avoidance): $(\max(0, s^{ij}_k(t)))$. Since the max function is used in the state violation

constraints, which means there is no net effect from the last term when inequality constraint isn't active; therefore, when such case is true for both predicted and actual trajectories, the problem is equivalent to the single-agent unconstrained trajectory planning.

$$\begin{aligned} J_e &= |J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) - J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i)| \\ &= C(\mathbf{x}_k^i, \mathbf{u}_k^i) - C(\mathbf{x}_k^i, \mathbf{u}_k^i) = 0 \end{aligned} \tag{5.10}$$

With the global estimation error on agent i's states $\mathbf{e}_k^i \leq \delta$, we have $\mathbf{e}^{ij} \leq \|\mathbf{e}_k^{1i}, \ldots, \mathbf{e}_k^{Ni}\|_\infty \leq \delta$, and this also implies the cumulative error by agent i is

$$\mathbf{E}_k^i = \|X_k^i - X_k\| = \|\mathbf{e}_k^{i1}, \ldots, \mathbf{e}_k^{iN}\| \leq \sqrt{N-1}\delta \tag{5.11}$$

, refer to (1.13,1.14) for the definition of X_k^i, X_k. By using Lipschitz condition, we have the following inequality:

$$\begin{aligned} J_e &= |J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) - J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i)| \\ &\leq L_{max}\|\mathbf{X}_k^i - \mathbf{X}_k\| \leq L_{max}\sqrt{N-1}\delta \end{aligned} \tag{5.12}$$

, which concludes the proof.

Assumption 5.2.2: Complementary slackness and exactness of nonlinear penalty function: We assume at each iteration k for agent $i \in V$, there always exists Lagrange multipliers $\gamma_{i,t} \in \mathbb{R}^n, \sigma_{i,t} \in \mathbb{R}$ and nonempty local optimal solution set $(\mathbf{x}_k^{i,*}, \mathbf{u}_k^{i,*}) \in \mathcal{X}_k^i \notin \emptyset$ (Slater's condition) such that Lagrangian is well-posed and is given by the follows (please note that $\mathbf{x}^{i,*}$ is the KKT points to (2.2), and $\mathbf{x}^{i*}$ is the optimal solution (partial minimum) to (2.7), which isn't necessary a KKT point to (2.2)):

$$\begin{aligned} &(\mathbf{x}_k^i, \mathbf{u}_k^i, u, v) = C(\mathbf{x}_k^i, \mathbf{u}_k^i) + \\ &\sum_{t=0}^{T}[\gamma_{i,t}^T(x^i(t+1) - f(x_k^i(t), u_k^i(t))) \\ &+ \sum_{j=1, j\neq i}^{N} \sigma_{i,t}(2r - \|x_k^i(t, 0:2) - x_k^j(t, 0:2)\|)] \end{aligned} \tag{5.13}$$

and that the following holds (complementary slackness):

$$x_k^{i,*}(t+1) - f(x_k^{i,*}(t), u_k^{i,*}(t)) = 0 \tag{5.14a}$$

$$\sigma_{i,t} = 0,\ 2r - \|x_k^{i,*}(t,0:2) - x_k^j(t,0:2)\| \ \leq 0 \tag{5.14b}$$

$$(2r - \|x_k^{i,*}(t,0:2) - x_k^j(t,0:2)\|) = 0,\ \sigma_{i,t} \geq 0 \tag{5.14c}$$

This assumption also has an implication that $s_k^{i,j}(T) = 0,\ \forall i,j \in V$, and the user has to ensure the terminal condition is feasible. If such a solution exists and satisfies both Slater's condition and complementary slackness, the duality gap is zero. Note that our nonlinear penalty terms in $J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i)$ is in the same form as the Lagrangian, which automatically ensures the strong duality to the (2.2) when the solution to (2.7) is feasible and satisfies the first order optimality conditions (KKT).

Assumption 5.2.3: Sufficient small trust region and consensus error: We require that the trust region is small enough such that "cross of agent" will never happen, which is equivalent to, say, once the individual solution is in the quasi-convex $\alpha-$sublevel set of J_k^i and is feasible in original problem: There exist $\kappa \geq k$ such that, $C_k^\alpha := \{(\mathbf{x}_k^i, \mathbf{u}_k^i, J_k^i) | J(\mathbf{x}_k^i) \leq \alpha, \alpha \geq 0,\ \forall i \in V\},\ \forall k \geq \kappa \geq 0$.

Assumption 5.2.3 is important in ensuring the stability of the low-order approximation convexification method, especially the use of ℓ_1 norm in the nonlinear penalty function.

Monotonic decrease of nonlinear penalty function before critical step:

Under Assumption 5.2.1, the cost of nonlinear penalty function decreases monotonically until reaching critical iteration $k \geq k_{crit}, k_{crit} \geq 0$, such that:

$$\begin{aligned} &J(\mathbf{x}_{k+1}^i, \mathbf{X}_{k+1}, \mathbf{u}_{k+1}^i) \leq J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i),\ \forall k \leq k_{crit} \\ &J(\mathbf{x}_{k+1}^i, \mathbf{X}_{k+1}, \mathbf{u}_{k+1}^i) \geq J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i), \\ &\frac{2\sqrt{N-1}L_{max}\delta}{\beta} \geq (\frac{1}{\lambda_2} - 1) \text{ for some } k \geq k_{crit} \end{aligned} \tag{5.15}$$

Again, refer to (1.13,1.14) for the definitions of $\mathbf{X}_k^i, \mathbf{X}_k$.

Proof: From the sufficient decreasing assumption to the local optimization problem, and let $\widetilde{\mathbf{x}_k^i} = \mathbf{x}_k^i + \mathbf{d}_k^i, \widetilde{\mathbf{X}}_k^i = \left[\mathbf{x}_k^{i1}, \ldots, \widetilde{\mathbf{x}}_k^{ii}, \ldots, \mathbf{x}_k^{iN}\right]^T, \widetilde{\mathbf{u}}_k^i = \mathbf{u}_k^i + \mathbf{w}_k^i$ denote variables for the intermediate step (see Algorithm 4), we have:

$$J(\widetilde{\mathbf{x}}_k^i, \widetilde{\mathbf{X}}_k^i, \widetilde{\mathbf{u}}_k^i) \leq J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) - \beta\|\mathbf{d}_k^i\| \tag{5.16}$$

Similar to $\widetilde{\mathbf{X}}_k^i$ we denote $\widetilde{\mathbf{X}}_k = \left[\mathbf{x}_k^1, \ldots, \widetilde{\mathbf{x}}_k^i, \ldots, \mathbf{x}_k^N\right]^T$, from (5.9) and assumption 5.2.1, we know the following inequalities:

$$J(\widetilde{\mathbf{x}}_k^i, \widetilde{\mathbf{X}}_k^i, \widetilde{\mathbf{u}}_k^i) \leq J(\widetilde{\mathbf{x}}_k^i, \widetilde{\mathbf{X}}_k, \widetilde{\mathbf{u}}_k^i) + L_{max}\sqrt{N-1}\delta \tag{5.17a}$$

$$J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i) + L_{max}\sqrt{N-1}\delta \geq J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k^i) \tag{5.17b}$$

$$\begin{aligned} J(\widetilde{\mathbf{x}}_k^i, \widetilde{\mathbf{X}}_k, \widetilde{\mathbf{u}}_k^i) &\leq J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i) + 2L_{max}\sqrt{N-1}\delta - \beta\|\mathbf{d}_k^i\| \\ &\leq J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i) + 2L_{max}\sqrt{N-1}\delta - \\ &\beta\delta(\frac{1}{\lambda_2} - 1) \end{aligned} \tag{5.18}$$

For $J(\widetilde{\mathbf{x}}_k^i, \widetilde{\mathbf{X}}_k, \widetilde{\mathbf{u}}_k^i) \leq J(\mathbf{x}_k^i, \mathbf{X}_k, \mathbf{u}_k^i)$, $2L_{max}\sqrt{N-1}\delta - \beta\|\mathbf{d}_k^i\| \leq 0$; therefore, the true cost decreases monotonically if $\frac{2\sqrt{N-1}L_{max}\delta}{\beta} \leq (\frac{1}{\lambda_2} - 1), \|\mathbf{d}_k^i\| = (\frac{1}{\lambda_2} - 1)$, which depends on the initial condition, will happen after some iterations $k \geq k_{crit} \geq 0$. This concludes our proof.

Note that the maximum tolerated consensus errors and trust region are uniform for all agents, as we initialize the trust region uniformly, and the trust region will update only after the nonlinear cost increase is detected by the local agent.

Comments on cost decreasing ratio $\frac{L_k}{\beta}$, connectivity λ_2, and consensus error bound δ: From (5.18), the main cause of the increase in the true cost is resulting from the consensus error, which makes agent sometimes predict their cost is decreasing while the true cost is actually raising. The immediate result of (5.18) is the stronger the network connectivity with smaller λ_2, the more unlikely the actual cost will increase. Also, the cost decreasing ratio $\frac{L_{max}}{\beta}$ can be understood as how much increase in the actual cost due to consensus error compared to how much cost is reduced in the estimated cost. Lastly, by choosing a small consensus error bound δ, we can ensure $\|\mathbf{d}_k^i\| = \delta(\frac{1}{\lambda_2} - 1)$ when $\|\mathbf{x}_k^i - z_k^i\| \geq r_k^i$. ($\mathbf{x}_k^i$ is not close to the stationary points) Note that, by using ℓ_1-norm in our trajectory cost and penalty cost, we can ensure a sufficient change in estimated cost with common solving techniques, including SCvx, as stated in assumption 5.2.1. Also, one should notice that when the graph is complete, there is no consensus error cost, which results in the monotonic decrease in the true cost, and the true cost is lower bounded by local minimum $J(\mathbf{x}^{i,*}, \mathbf{X}_k, \mathbf{u}^{i,*})$ at its corresponding partial minimum; therefore, we have the strong convergence of solutions to at least partial minimum of for each agent with a complete graph. Please note that the KKT

points or partial minimum defined across the work are defined for the local problem, which in general won't be the same KKT point or partial minimum for the centralized problem when we stack all variables together.

Implication of critical step k_{crit}: assume the trajectory is feasible, $k = k_{crit}$ means the followings: k^i_{crit} is the first iteration that the cost increase due to consensus error is more than the cost decrease from the local optimization.

If the solution is dynamically feasible but does not satisfy the collision constraints, the true cost of each agent $J(\mathbf{x}^i_k, \mathbf{X}_k, \mathbf{u}^i_k) = \sum_{t=0}^{T} \lambda \max(0, s^{ij}_k(t)) + C(\mathbf{x}^i_k, \mathbf{u}^i_k)$, we will establish that the cost incurred by the consensus error $S^i_k = \sum_{t=0}^{T} \sum_{j=1 j\neq i}^{N} \lambda \max(0, s^{ij}_k(t))$ is bounded and conditions under which this cost will decrease after the critical step k_{crit}. We will also consider the δ^i_k individually from now on, as the trust regions aren't likely to remain uniform when $k \geq k_{crit}$.

Detectable cost increase due to consensus error:

Equipped with assumption 5.2.2 and denote

$$s^{ij}_k(t)^T = 2r - \|x^i_k(t, 0:2) - x^j_k(t, 0:2)\|,\ S^{i,T}_k = \sum_{t=0}^{T} \sum_{j=1, j\neq i}^{N} \lambda \max(0, s^{ij}_k(t)^T) \tag{5.19}$$

(T denotes true data) the cost incurred by the consensus error, we claim that if $\delta > 0$, there exist finite $0 \leq k < \infty$ such that, $J(\mathbf{x}^i_k, \mathbf{X}^i_{k+1}, \mathbf{u}^{i,*}_k) > J(\mathbf{x}^i_k, \mathbf{X}^i_k, \mathbf{u}^{i,*}_k), \forall i \in C := \{i, |S^{i,T}_k > 0\}$. **Please pay attention to the subscript.**

Proof: By using contradiction, we first assume an opposite case that $J(\mathbf{x}^i_k, \mathbf{X}^i_{k+1}, \mathbf{u}^i_k) \leq J(\mathbf{x}^i_k, \mathbf{X}^i_k, \mathbf{u}_k),\ \forall k \in [0, \infty)$, and each agent will reach local minimizers in finite k such that $J(\mathbf{x}^{i*}_k, \mathbf{X}^i_{k+1}, \mathbf{u}^{i,*}_k) = C(\mathbf{x}^{i,*}_k, \mathbf{u}^{i,*}_k) + S^i_k,\ S^i_k = 0$. Since each agent is at their local minimum, the solution won't change in the next iteration, $\mathbf{x}^{i,*}_k = \mathbf{x}^{i,*}_{k+1}, \mathbf{u}^{i,*}_k = \mathbf{u}^{i*}_{k+1},\ \forall i \in V$, which is equivalent to $\|d^i_k\| = 0,\ \forall i$ (all agents are at local optimal conditions). From Assumption 5.2.3 (complementary slackness) we know that at optimal, if the inequality constraint is active, $2r = \|x^{i,*}_k(t, 0:2) - x^{ij}_k(t, 0:2)\|,\ \forall t \in [0, T]$. Also, from the previous results, the consensus error $\|x^{ij}_k(t, 0:2) - x^j_k(t, 0:2)\| \to 0, \forall i$ if $\|d^i_\kappa\| = 0,\ \forall \kappa \geq k$. Since $S^{i,T}_k > 0$, $2r - \|x^{i,*}_k(t, 0:2) - x^j_k(t, 0:2)\| > 0$ for some t, there exists $k \leq \kappa < \infty$ such that

$2r - \|x_\kappa^{i,*}(t, 0:2) - x_\kappa^{ij}(t, 0:2)\| > 0$ and

$$\begin{aligned} J(\mathbf{x}_{\kappa-1}^{i,*}, \mathbf{X}_\kappa^{i,*}, \mathbf{u}_{\kappa-1}^{i,*}) &= C(\mathbf{x}_{\kappa-1}^{i,*}, \mathbf{u}_{\kappa-1}^{i,*}) + S_k^i \\ > J(\mathbf{x}_{\kappa-1}^{i,*}, \mathbf{X}_{\kappa-1}^{i,*}, \mathbf{u}_{\kappa-1}^{i,*}) &= C(\mathbf{x}_{\kappa-1}^{i,*}, \mathbf{u}_{\kappa-1}^{i,*}) + 0 \end{aligned} \tag{5.20}$$

. If we replace $\kappa - 1$ with some k with the same condition, then we have $J(\mathbf{x}_k^i, \mathbf{X}_{k+1}^i, \mathbf{u}_k^i) > J(\mathbf{x}_k^i, \mathbf{X}_k^i, \mathbf{u}_k)$, which contradicts the assumption we made earlier; this completes the proof that there always exist finite k for agent to detect the increase in the cost associated with the consensus error.

Importance of detectable cost rise: Consider $\|\mathbf{x}_k^i - \mathbf{x}_k^{i,*}\| \geq r_k^i$, we have $\|\mathbf{d}_k^i\| = r_k^i$. Under this condition, we have an equality relationship between the current consensus error and trust region such that $r_k^i = \delta_k^i(\frac{1}{\lambda_2} - 1)$. Whenever the cost increase is detected, $r_{k+1}^i = \frac{r_k^i}{\omega}$, and $\delta_{k+1}^i = \frac{\delta_k^i}{\omega}$. With the mechanism to detect and reduce the consensus error once the consensus error cost increase is detected, the consensus error among agents with active safe constraints will decrease over iteration; furthermore, when the trajectory is feasible for local problem (2.3), the cost associated with consensus error will converge a small chosen value $\delta_{min} > 0$.

In the previous section, we claimed that the error for the nonlinear penalty function is bounded in terms of the cyber layer (graph connectivity); we are also interested in its boundness in terms of the physical characteristics (diameter) of each agent to better understand graph connectivity and its influence on the actual physical system.

Assumption 5.2.4: Locally feasible trajectory for all agents after finite iteration $k_{feasible}$: Even sometimes, we start the algorithm with infeasible trajectories for agents $S_k^i \geq 0, x_k^i(t+1) - f(x_k^i(t), u_k^i(t)) \neq 0$ for some $t \in [0, T]$, we assume, after finite iterations, there exists $k_{feasible} \geq 0$, such that the trajectory perceived by the agent with local information will be locally feasible for problem (2.3) for all $k \geq k_{feasible}$ such that followings are true: $S_k^i = 0, x_k^i(t+1) - f(x_k^i(t), u_k^i(t)) = 0, x_k^i(t+1) = A_d^i(t)x_k^i(t) + B_d^i(t)u_k^i(t)\ \forall t \in [0, T]$. This assumption can be extended to many general SCP solvers, including SCVX with proper initialization [16]. This assumption also implies after sufficient iterations, $s_k^{i,j}(0) = 0,\ \forall i, j \in V$.

Assumption 5.2.4 is a common assumption used in many works involving constraint

relaxation methods[8, 10], which introduce sets of stationary and partial minimum points that are not feasible to (2.7), and is illustrated by Figure 5.1. Also, note that all the KKT points and stationary points belong to partial minima, which implies that if we initialize our trajectory properly or in the neighborhood of local minima, our algorithm will converge to the local minima, provided the graph is complete.

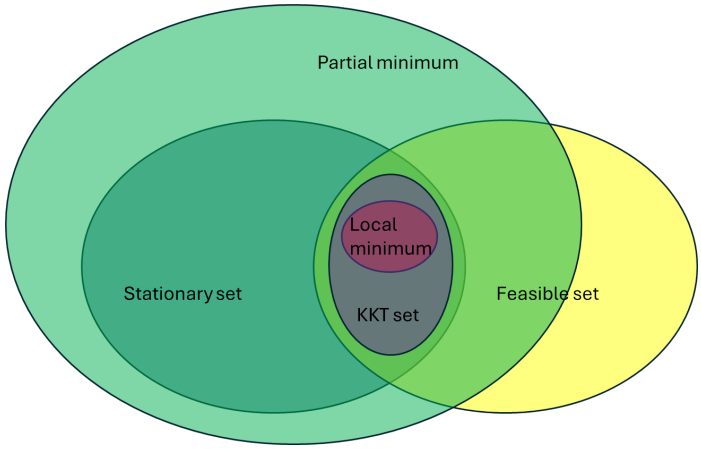

Figure 5.1: Illustration showing the sets of feasible solutions to (2.3), stationary and partial minimum sets to (2.7), and KKT points/ local minimizers to both (2.3,2.7)

Assumption 5.2.5: Sufficient small final consensus error for colliding agents: Assume a small final consensus error between the agents with active collision constraints, such that the minimum required consensus error tolerance chosen by the user satisfies $\delta_{min} \leq \sigma r(N-1)(T-2)$, where $0 < \sigma \leq 1$ is the user-defined parameter for determining how much of collision constraints can be violated.

Bounded cost associated with consensus error:

Under assumptions 5.2.2, 5.2.4 and 5.2.5, for all iterations $k \geq 0$, the consensus error costs $S_k^{i,T}, S_k^i$ for both true trajectory and local estimation are bounded above by $2r\lambda(N-1)(T-2)$, and in the finite iteration $\kappa \geq k >> 0$, such cost will have the bound given by:

$$S_\kappa^{i,T} \leq \delta_{min}\lambda(N-1)(T-2) \leq \sigma r(N-1)(T-2),\ \forall \kappa \geq k, \tag{5.21}$$

Proof: Recall that $S_k^i = \sum_{t=0}^{T}\sum_{j=1 j\neq i}^{N} \lambda \max(0, 2r - \|x_k^i(t) - x_k^{ij}(t)\|)$, by setting $x_k^i(t) -$

$x_k^{ij}(t) = 0,\ \forall j \neq i$, we have $\sup_{\mathbf{x}_k^i}(S_k^i) = 2r\lambda(N-1)(T-2)$.

Now, consider the following figure with the assumption that both agents have the same consensus error $\Delta x = \|x^2(t) - x^{12}(t)\| = \|x^1(t) - x^{21}(t)\|$ at time $t \geq 0$, and we let $\Delta x \leq r$. Note that from the local agent's perspective, they don't know the collision and the trajectory is feasible for both agents based on their estimates. We consider a special case where

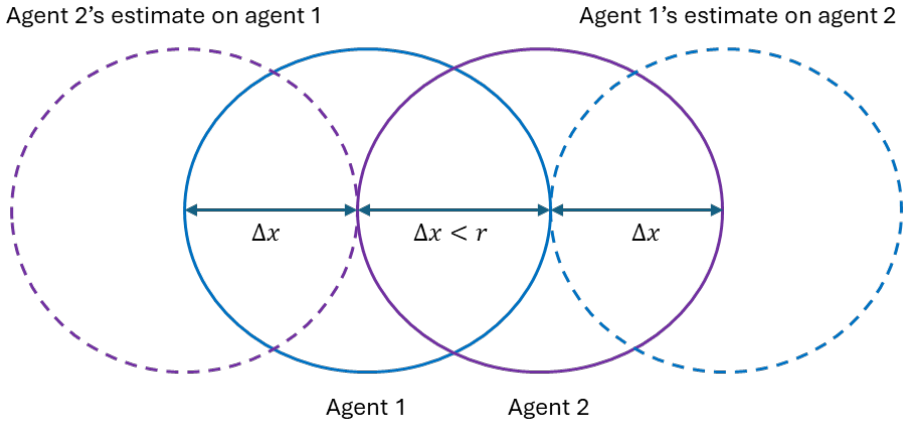

Figure 5.2: Illustration of agent collision due to the consensus error

$t = [0, 1, 2]$, and $x(t = 1)$ for both agents, which means there are only three time-steps in the whole trajectory with $t = 0, t = 2$ being the initial and final time steps. Since the maximum-smallest consensus error is bounded by $\mathbf{e}_k^1 = \mathbf{e}_k^2 \leq \delta_{min}$, we have:

$$\begin{aligned} &\mathbf{e}_k^1 = \|\mathbf{x}_k^2 - \mathbf{x}_k^{12}\| = \|x_k^2(1) - x_k^{12}(1)\| = \\ &\mathbf{e}_k^2 = \|\mathbf{x}_k^1 - \mathbf{x}_k^{21}\| = \|x_k^1(1) - x_k^{21}(1)\| \leq \\ &\delta_{min} \leq \sigma \end{aligned} \tag{5.22}$$

With the above arguments, we have $\sup_{\mathbf{x}_k^1}(S_k^1) = \sup_{\mathbf{x}_k^2}(S_k^2) = \sigma r$, and by extending this to the general case with T time steps and N agents $\sup_{\mathbf{x}_k^i}(S_k^i) = \sigma r(N-1)(T-2)$, which completes the proof to (5.21).

Strong convergence with diminishing trust region design: One undesired behavior with the current design is the lack of convergence guarantee, which can be addressed if one removes the algorithm's δ_{min} condition such that $r_k^i, \delta_k^i \to 0$ as $k \to \infty,\ \forall i \in C$.

Chapter 6

NUMERICAL SIMULATION AND HARDWARE EXPERIMENTS RESULTS

Numerical simulation with a few hardware experiments will be given to verify our claims, and also demonstrate the computational advantages with our novel SVM reformulation method.

6.1 Johnny robots

In this section, we provide a brief overview of the mathematical models for our in-house-built Johnny robots and the system architectures.

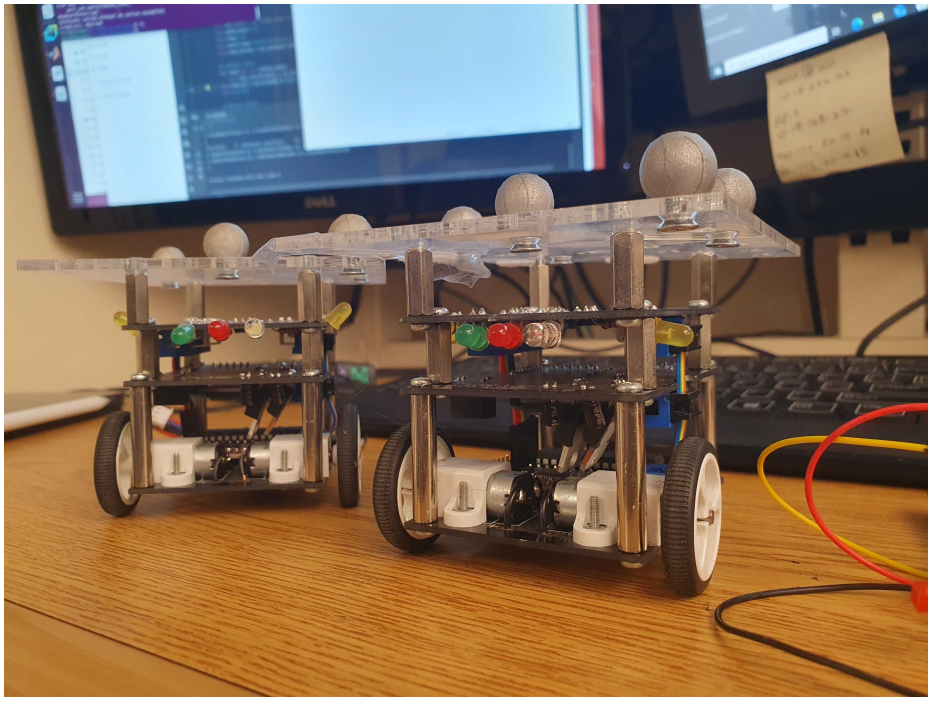

Figure 6.1: Two Johnny robots

6.1.1 Unicycle model

The unicycle dynamics is selected as the representative nonlinear model, as many ground robots have a similar dynamics, including our Johnny robots. The unicycle dynamics are represented by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{6.1}$$

where $x, y, \theta \in \mathbb{R}$ represents the position in $\mathbb{R}^2$ and the heading angle, respectively. Control inputs $v, \omega \in \mathbb{R}$ are linear velocity and angular rate along z-axis, respectively. We also have some additional constraints on controls and states due to physical limits of Johnny robots, which requires $\omega \leq 3\ (rad/sec), 0 \leq v \leq 10\ (cm/sec)$.

We will make use of SCvx to handle the nonlinearity of the model with the discrete linear model given as follows:

$$\begin{bmatrix} x(t+1) \\ y(t+1) \\ \theta(t+1) \end{bmatrix} = \begin{bmatrix} 1, & 0, & -dtv\sin\theta \\ 0, & 1, & dtv\cos\theta \\ 0, & 0, & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} dt\cos\theta, & -\frac{1}{2}dt^2\sin\theta \\ dt\sin\theta, & -\frac{1}{2}dt^2\cos\theta \\ 0, & dt \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{6.2}$$

For the actual implementation, we used a special treatment of the unicycle model by approximating the double integrator to the unicycle model through adding additional constraints for faster computation.

6.1.2 System architectures

The entire control and guidance system for Johnny robots is composed of three parts: 1. High level optimization that take the desired user inputs to generate reference trajectories for each robot to follow; 2. guidance controller that convert the time-parameterized reference data from optimization to the desired angular and linear velocities, 3. low level velocity controller that uses PID to convert the desired velocity command to the motor signals to drive the wheels.

In the hardware perspective, the entire system includes: 1. computer A hosting the optimization program , 2. computer B running the velocity controller and receiving the velocity command from the computer A with states data coming from Vicon, 3. Vicon detecting the states and sending them to computer B, 4. Johnny robots receiving the motor command from the computer B. The overall system flowchart is given below.

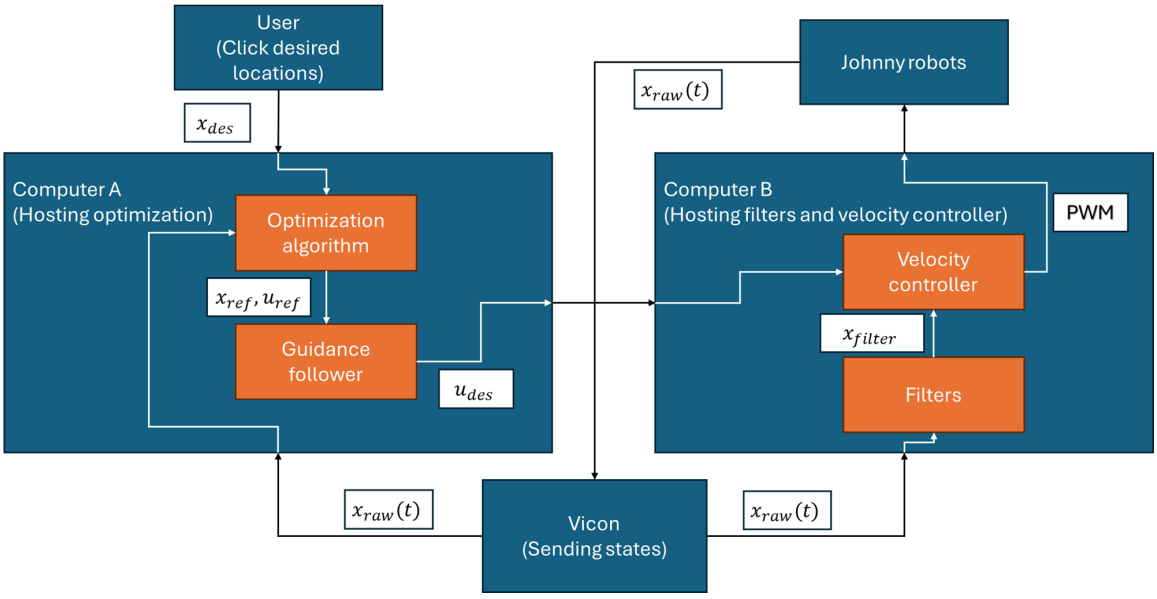

Figure 6.2: Johnny system architecture

6.2 Single agent trajectory planning

In this section, we will provide comprehensive numerical simulations and hardware experiments to demonstrate the unique features of our SVM reformulation method.

6.2.1 Numerical simulation

We begin with the numerical simulation with the single-agent case, and all the constraints are based on the actual Johnny robots. The same algorithm with the same constraints will also be used on the hardware simulation. We present the simulation results with two cases, results after just one iteration, and results after the algorithm fully converges(ten

iterations). From the figures in this section, it's evident that the fully converged solution will provide much smoother control input compared to the first iteration results. Apart from the SVM reformulation, we also provided the naive linearization version for comparison. Unlike naively linearizing the safe constraints and solving with the SCP method, the SVM reformulation can provide a guaranteed safe trajectory even if one initializes the trajectory passing through the obstacles. With the author's regular laptop equipped with a Intel i7-13900 CPU, performing ten iterations takes, in general, less than a second after the initial compilation. Lastly, the naive linearization approach not only violates the safe constraints but also requires more iterations to converge, and the resulting trajectory drastically differs from that of the SVM reformulation.

Table 6.1: Parameters used in the numerical simulation

	Parameters
Initial position	$\Big[0.2, 0\Big]$ (m)
Terminal position	$\Big[0.8, 0.8\Big]$ (m)
Initialization	straight line (infeasible)
Max linear velocity	10 cm/s
Max angular velocity	3 rad/s
SCP initial trust region	1 (m)
SCP trust region reduction ratio	1.1
Iteration to converge (SVM)	2
Iteration to converge (Pure linearization)	8
Total control $\|\mathbf{u}\|_2$ (SVM)	0.02095
Total control $\|\mathbf{u}\|_2$ (Pure linearization)	0.02135

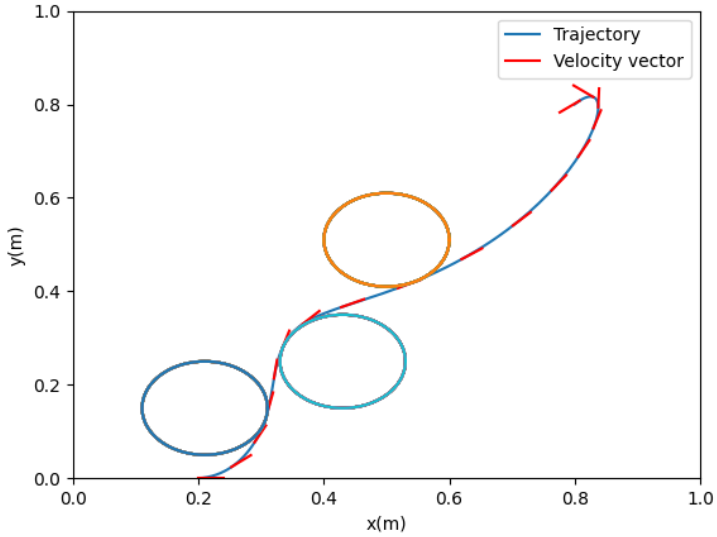

Figure 6.3: Trajectory after one iteration with SVM reformulation

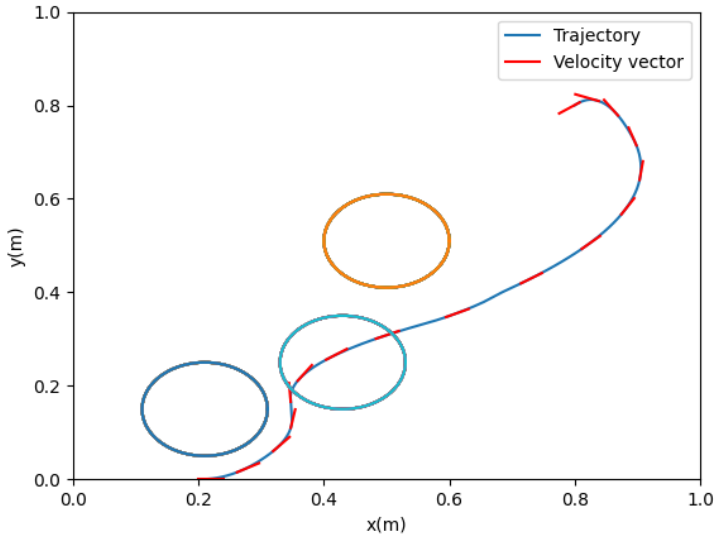

Figure 6.4: Trajectory after one iteration with only linearized safe constraints

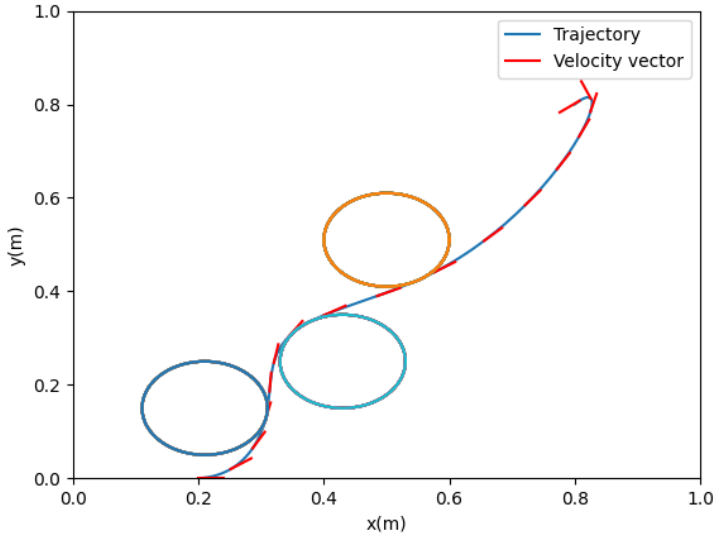

Figure 6.5: Trajectory after ten iterations with SVM reformulation

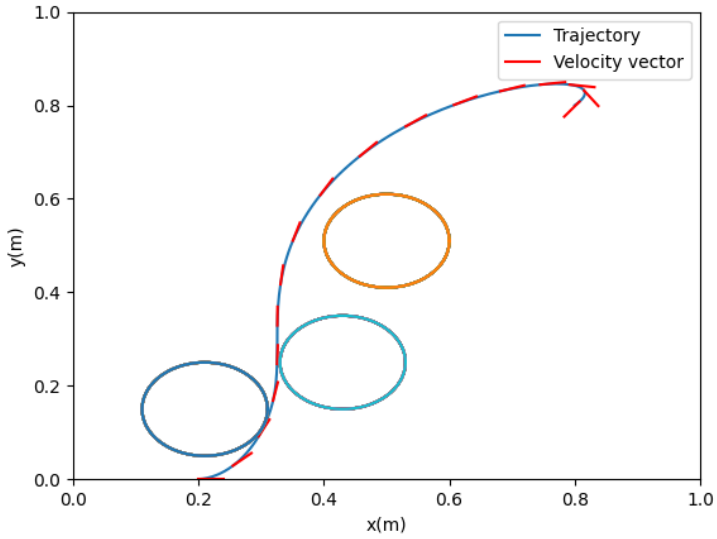

Figure 6.6: Trajectory after ten iterations with only linearized safe constraints

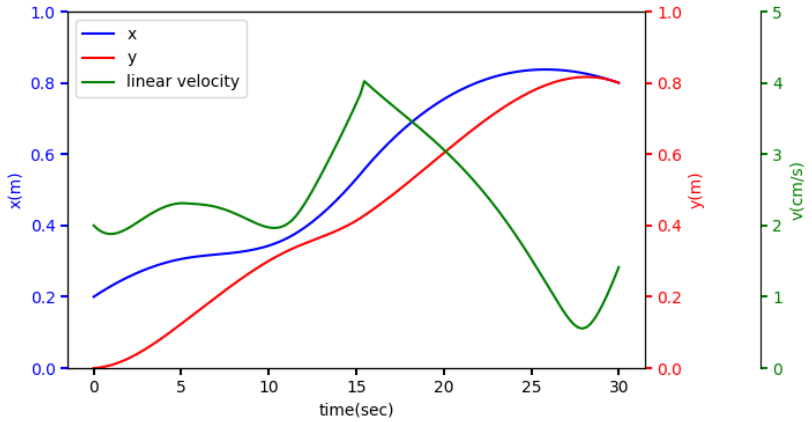

Figure 6.7: Agent states after one iteration with SVM reformulation

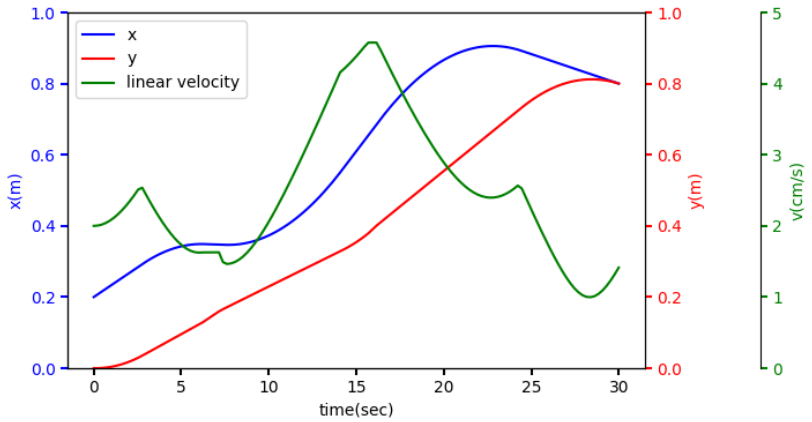

Figure 6.8: Agent states after one iteration with only linearized safe constraints

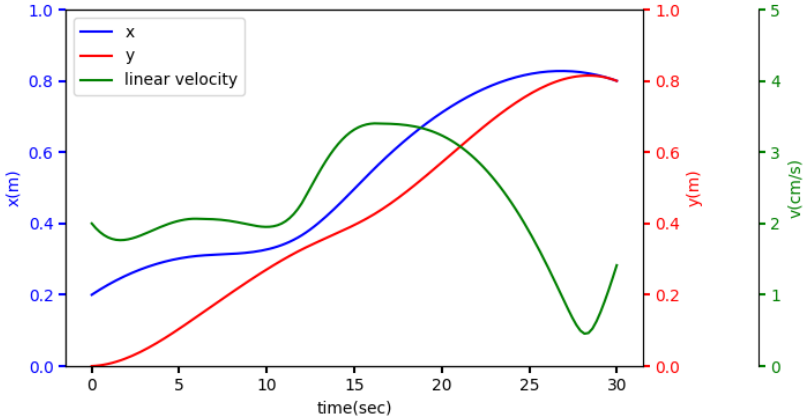

Figure 6.9: Agent states after ten iterations with SVM reformulation

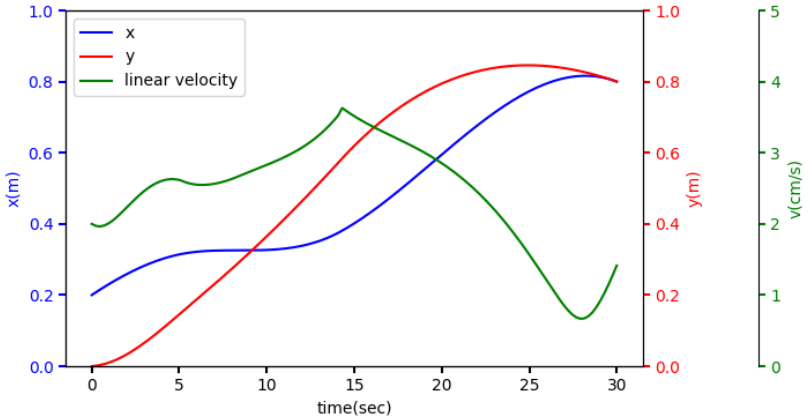

Figure 6.10: Agent states after ten iterations with only linearized safe constraints

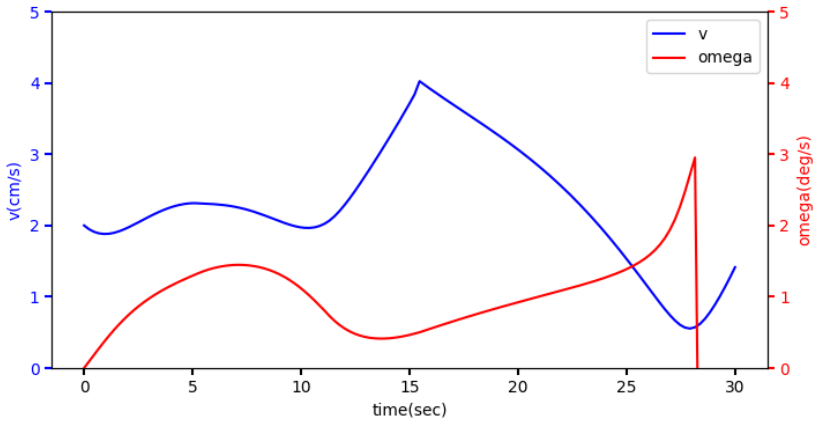

Figure 6.11: Control inputs after one iteration with SVM reformulation

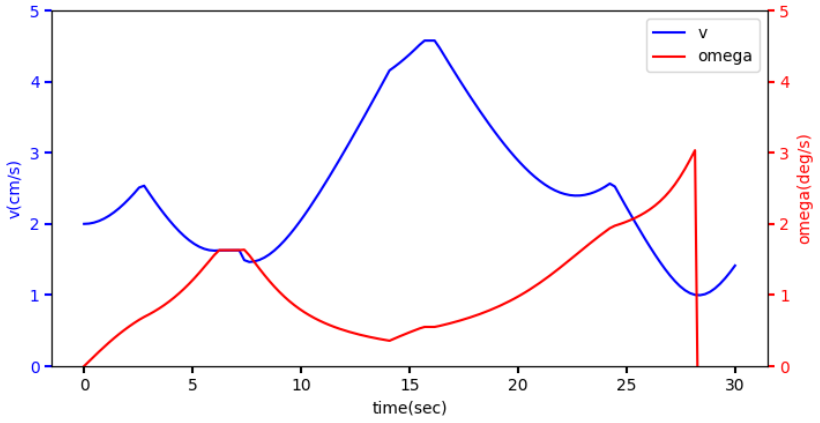

Figure 6.12: Control inputs after one iteration with only linearized safe constraints

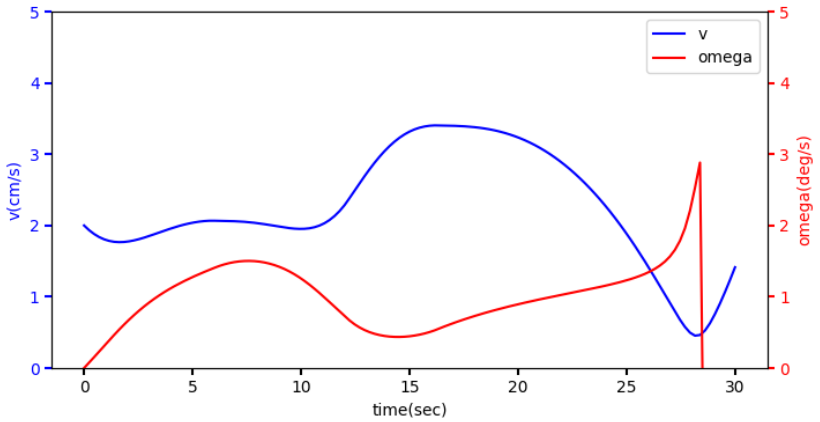

Figure 6.13: Control inputs after ten iterations with SVM reformulation

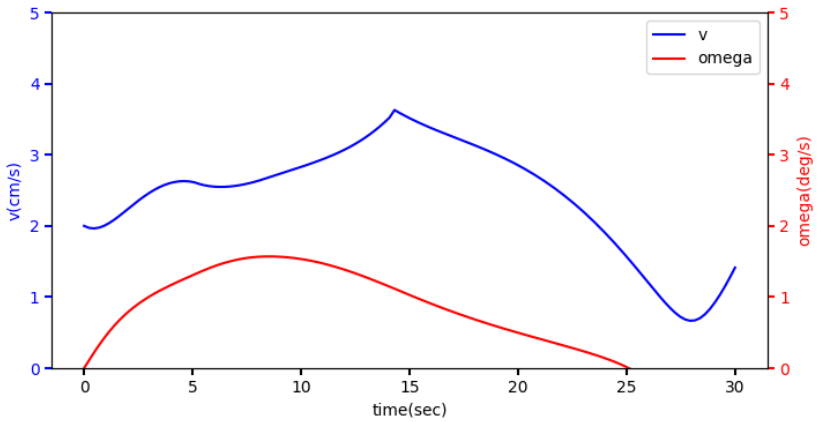

Figure 6.14: Control inputs after ten iterations with only linearized safe constraints

6.3 Decentralized multiagent trajectory planning

In this section, we will provide both numerical and experimental results to verify the proposed algorithm. We also demonstrate how the graph types and the number of consensus iterations affect the algorithm's behavior and consensus errors.

6.3.1 Ten agents case with C10 graph

To start, we give the numerical simulation for the ten-agent case with the C10 graph. The results shown below are obtained after 20 iterations, which can be solved in 3 seconds if multiprocessing is enabled or with multiagent to solve in parallel. Note that the length scale and linear velocity limit are increased for demonstration purposes.

Table 6.2: Parameters used in the numerical simulation for C10 graph

	Parameters
Graph type	C10
Max linear velocity	2 m/s
Max angular velocity	3 rad/s
SCP initial trust region	1 (m)
SCP trust region reduction ratio	1.1
Consensus initial trust region	5 (m)
Consensus trust region reduction ratio	1.1
Number of consensus iterations	5

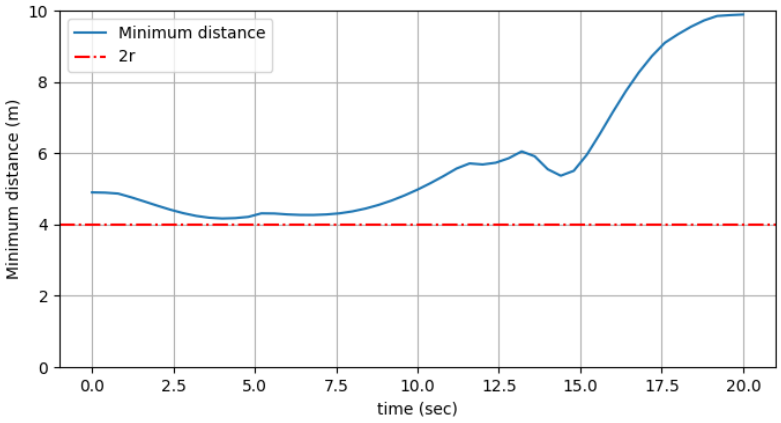

Figure 6.15: Minimum distance between agents after 20 iterations with ten agents

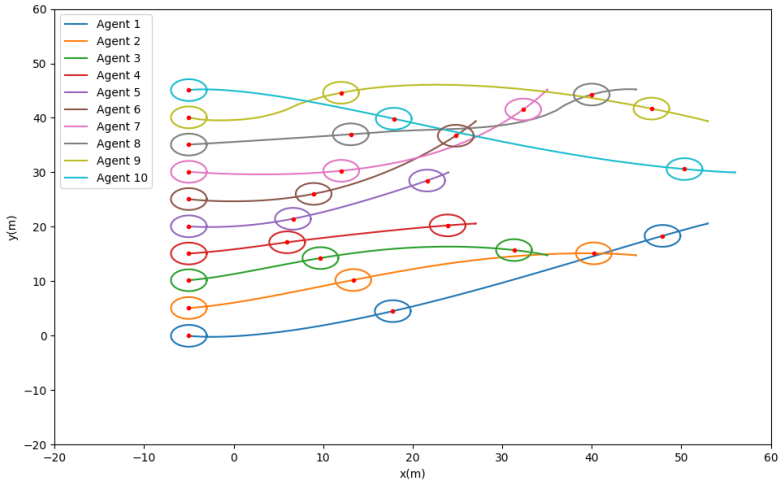

Figure 6.16: Trajectories of each agent after 20 iterations with ten agents to form a circular pattern

6.3.2 Numerical simulation with C4 graph

In this numerical simulation part, we intentionally increase the length scales to better show the effects of consensus errors on the overall convergence. Except for the increased linear

velocity constraint, all other constraints still remain the same as in the previous sections.

Table 6.3: Parameters used in the numerical simulation for C4 graph

	Parameters
Graph type	C4
Initial position	$\Big[0, 0; 0, 5; 0, 10; 0, 15\Big]$ (m)
Terminal position	$\Big[15, 15; 15, 10; 15, 5; 15, 0\Big]$ (m)
Max linear velocity	1 m/s
Max angular velocity	3 rad/s
SCP initial trust region	1 (m)
SCP trust region reduction ratio	1.1
Consensus initial trust region	5 (m)
Consensus trust region reduction ratio	1.1
Number of consensus iterations	1(case 1)/5(case 2)

6.3.3 General C4 cases

We present the four-agent cases with a circular C4 graph to show how consensus error can drastically change the converged solution even if all other parameters are fixed. By comparing the estimated cost of each case, one can observe that there is a large delay in the cost estimation for case 1, which is caused by the lack of sufficient communication between agents. As the iteration continue, the consensus error will be brought down as the agents will eventually detect the increase of nonlinear cost and reduce the consensus trust region, which, in turn, further reduces the consensus errors. Also, for case 2, the convergence is obtained not by forcing the trust region to decrease, which is reflected by its nearly zero consensus error after iteration 42. Trajectory evolutions are also heavily affected by the consensus error, as trajectories from the two cases are completely different. It is worth noting that performing such optimization when using multiprocessing usually takes less than five seconds for the entire fifty iterations. With more frequent communication, as in

case 2, it is possible to generate safe and feasible trajectories in a second.

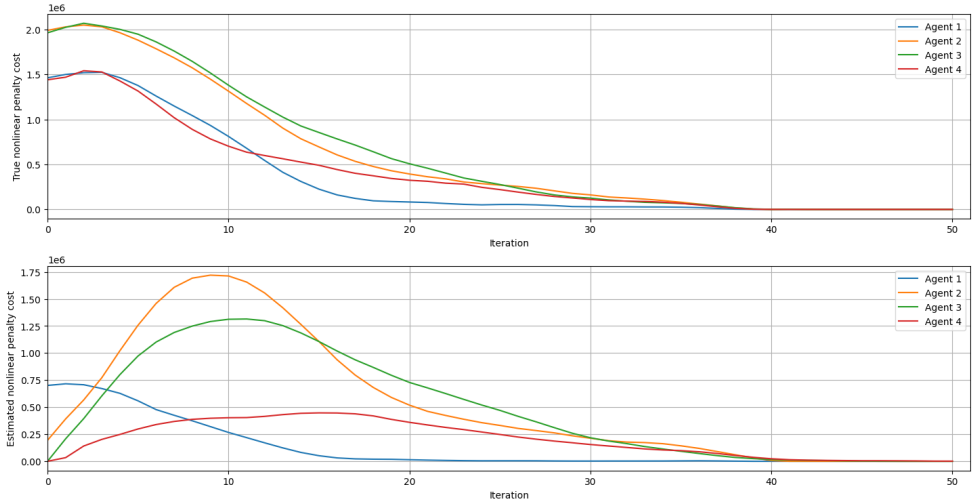

Figure 6.17: True and estimated nonlinear penalties for case 1

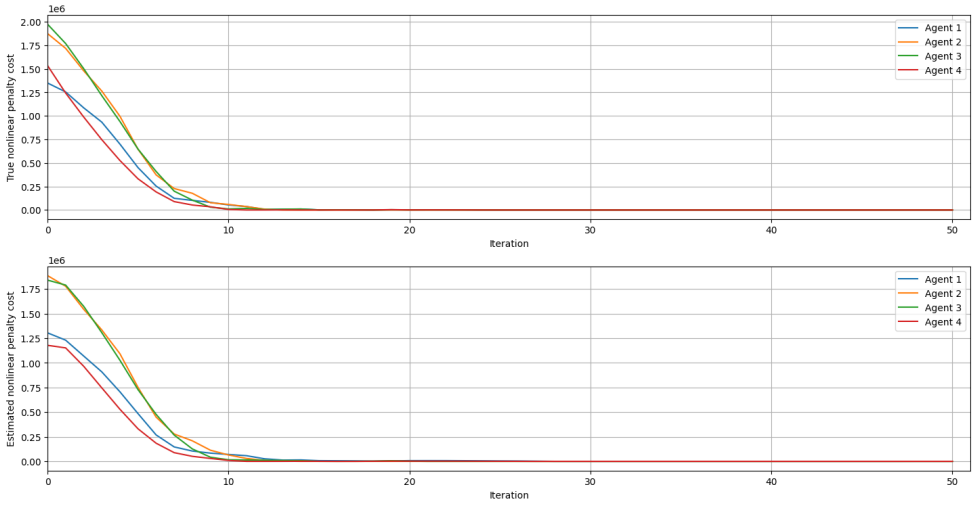

Figure 6.18: True and estimated nonlinear penalties for case 2

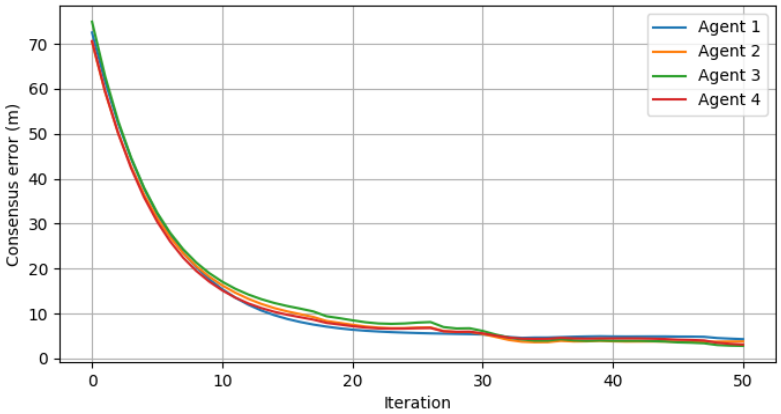

Figure 6.19: Consensus errors for case 1

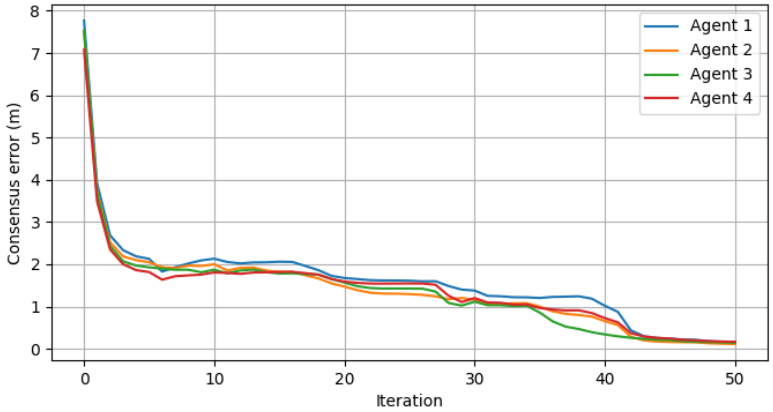

Figure 6.20: Consensus errors for case 2

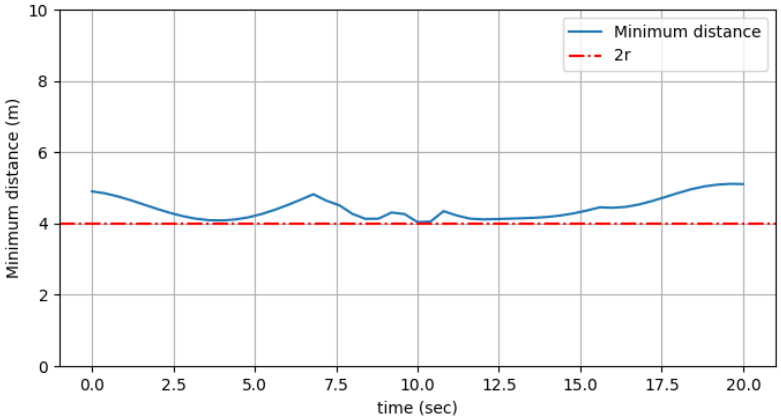

Figure 6.21: Minimum distance between agents for case 1

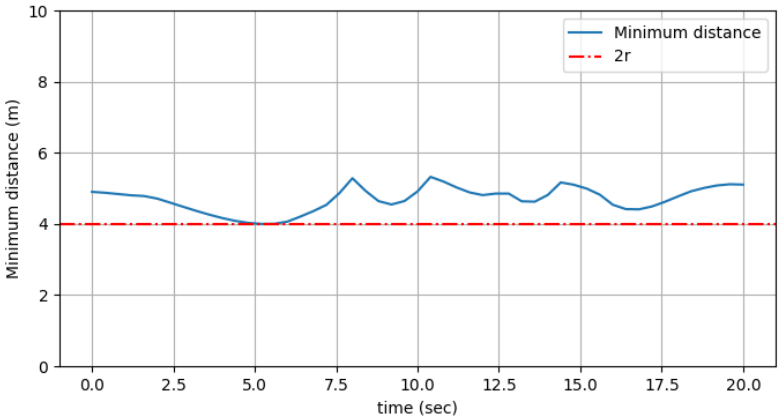

Figure 6.22: Minimum distance between agents for case 2

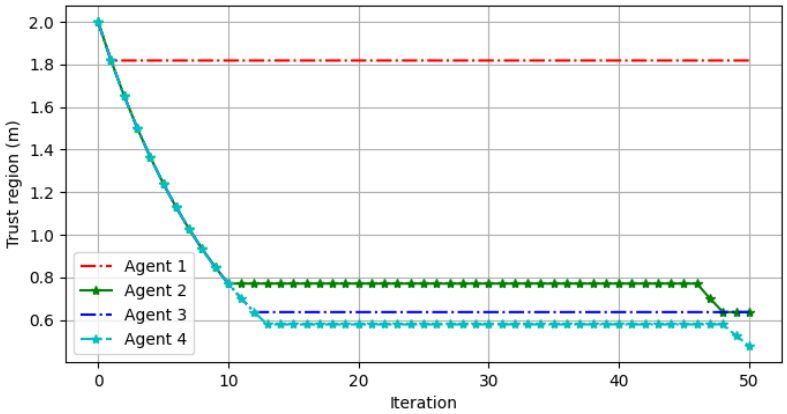

Figure 6.23: Trust regions of each agent for case 1

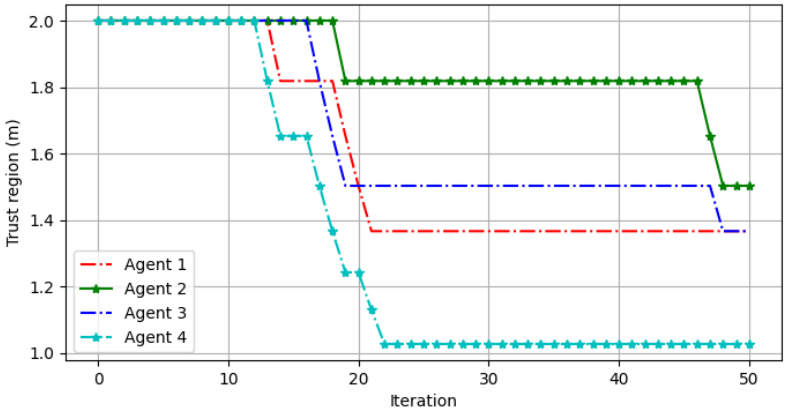

Figure 6.24: Trust regions of each agent for case 2

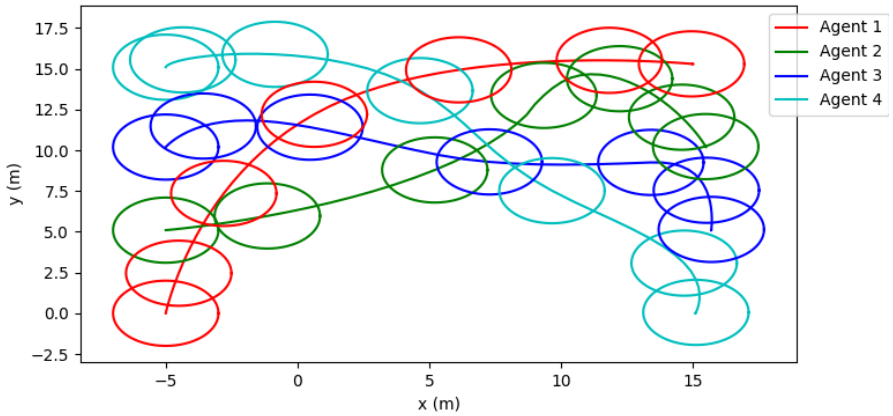

Figure 6.25: Trajectories of each agent after 50 iterations for case 1

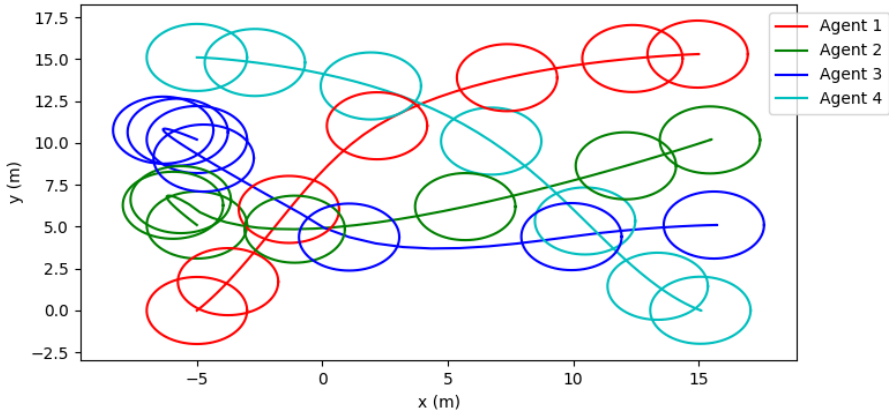

Figure 6.26: Trajectories of each agent after 50 iterations for case 2

6.3.4 Special case: Decentralized SCvx method with complete graph

We present the special case where there is no consensus error, and the resulting nonlinear penalties converge well in ten iterations. The evolution of trajectories also stops after 30 iterations, which can be seen from Fig 6.29. The converged trajectories also differs significantly from that of C4 case with five consensus iterations.

Table 6.4: Parameters used in the numerical simulation for C4 graph

	Parameters
Graph type	K4
Initial position	$\Big[0, 0; 0, 5; 0, 10; 0, 15\Big]$ (m)
Terminal position	$\Big[15, 15; 15, 10; 15, 5; 15, 0\Big]$ (m)
Max linear velocity	1 m/s
Max angular velocity	3 rad/s
SCP initial trust region	1 (m)
SCP trust region reduction ratio	1.1
Consensus initial trust region	5 (m)
Consensus trust region reduction ratio	1.1
Number of consensus iterations	1

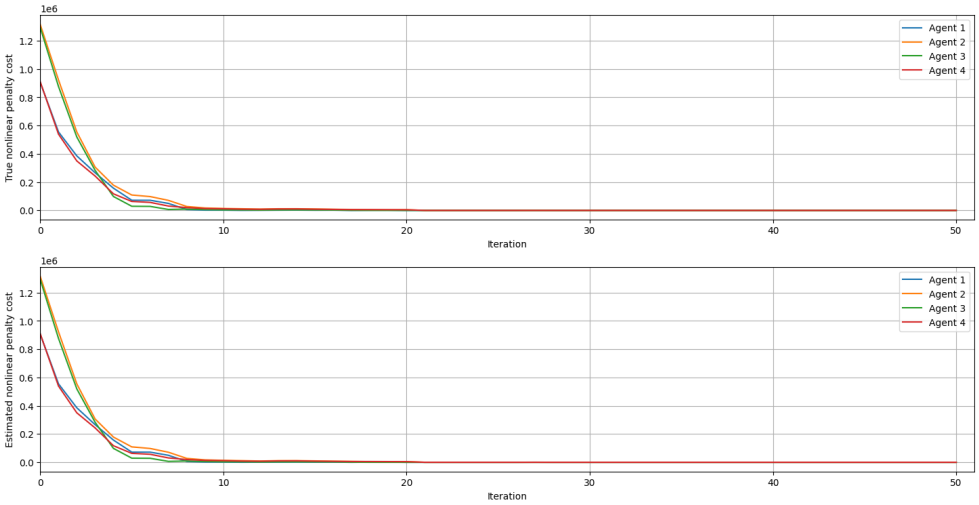

Figure 6.27: True and estimated nonlinear penalties for K4 graph

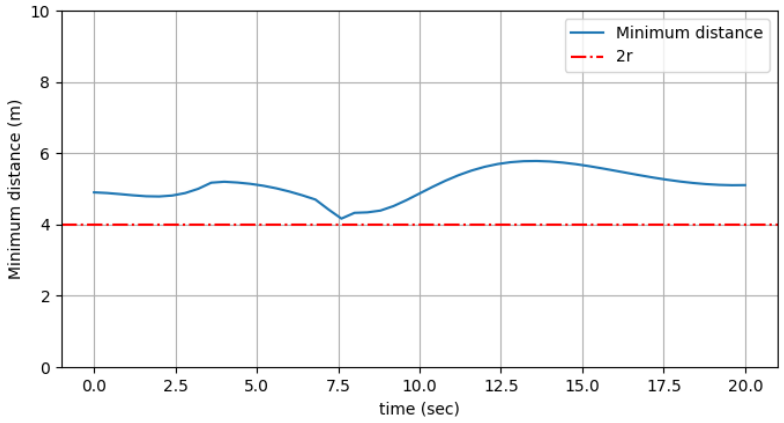

Figure 6.28: Consensus errors for K4 graph

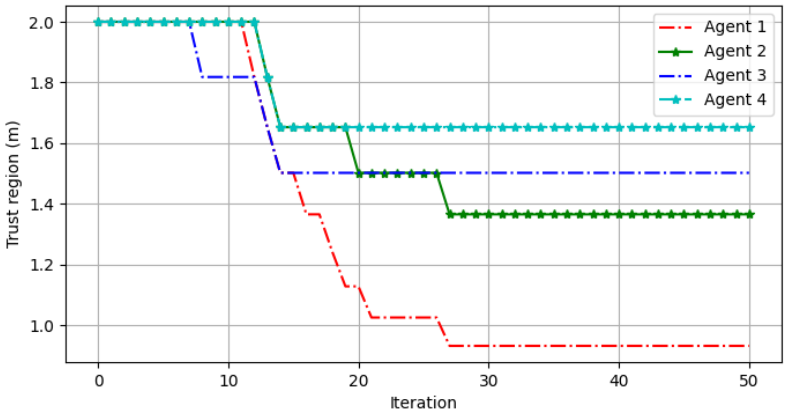

Figure 6.29: Minimum distance between agents for K4 graph

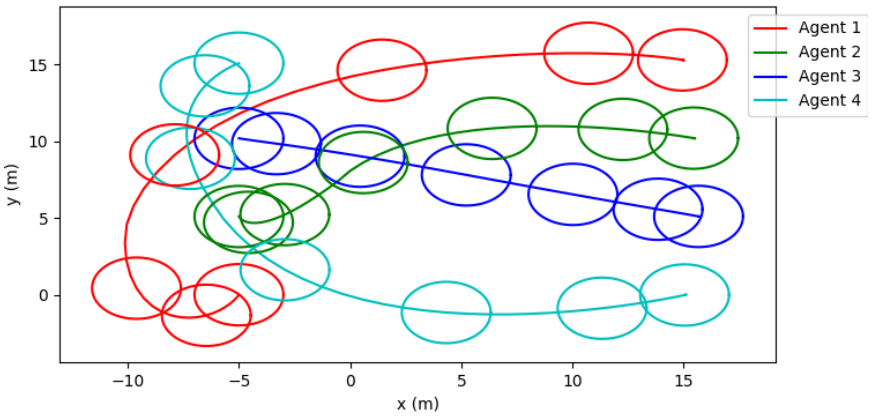

Figure 6.30: Trajectories of each agent after 50 iterations for case 2

6.3.5 Hardware experiments

In the hardware experiment, we included three Johnny robots starting at three random non-colliding positions with the target positions chosen by clicking on the screen. In the following two cases, we deliberately choose the crossing paths to test whether our algorithm can generate a safe path. Experiments show in most cases, safe trajectories can be generated in a relatively short time. However, there exists no solutions in some test runs when following conditions happens: 1. the desired positions are overlapped (no feasible solution), 2. during the algorithm iteration, some agents will "push" other agents away too much that no feasible solution exists for the given terminal time, which can be addressed by increasing terminal time.

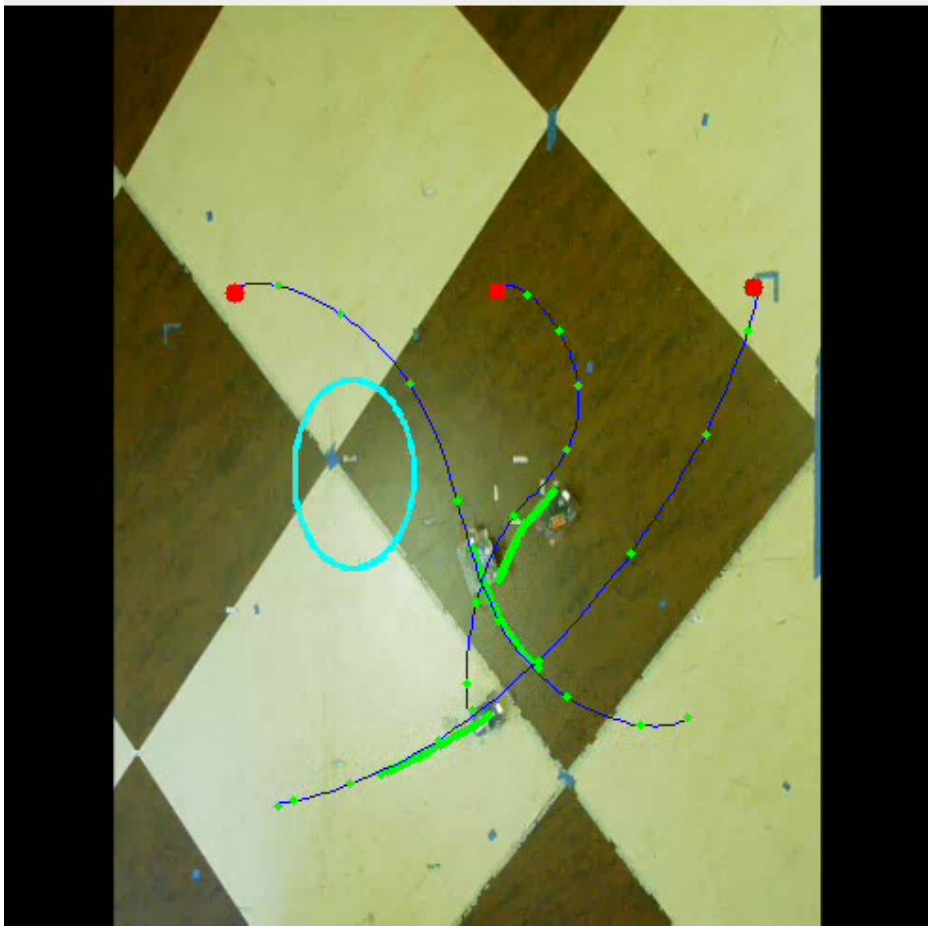

Figure 6.31: Hardware experiment with three robots and one circular obstacle case one

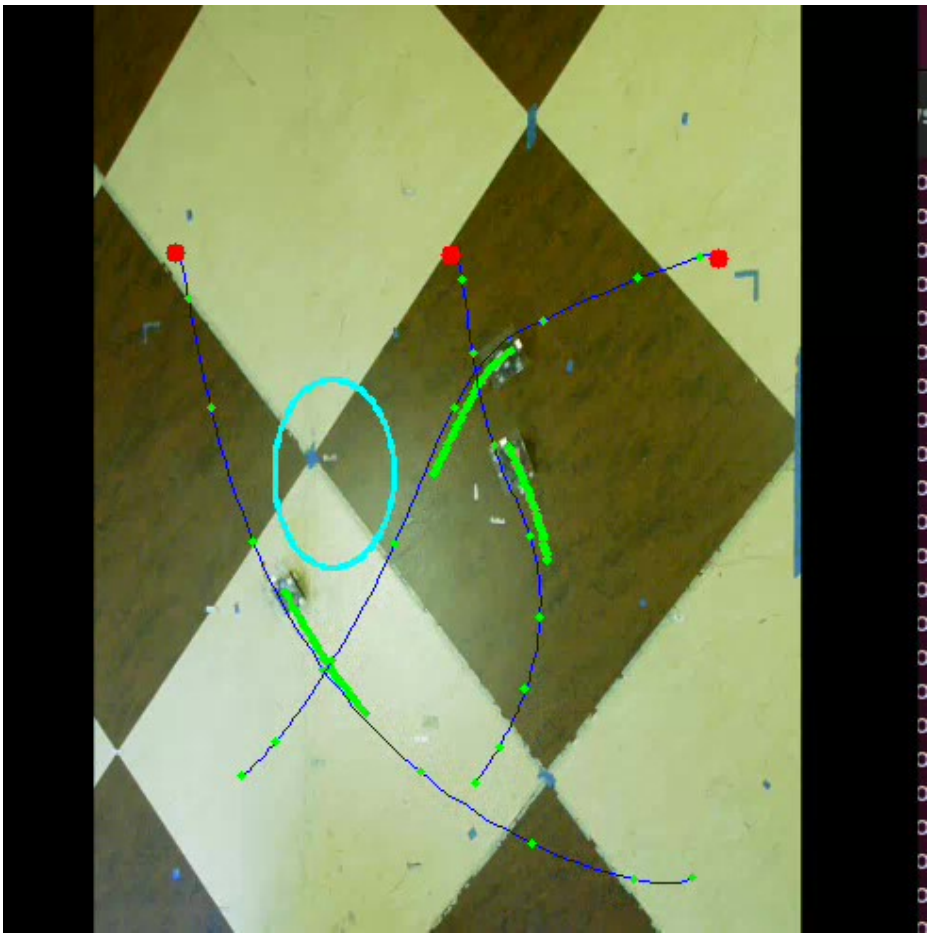

Figure 6.32: Hardware experiment with three robots and one circular obstacle case two

Chapter 7

CONCLUSIONS

In this work, we presented a scalable distributed trajectory optimization supported by the SCP method with SVM reformulation to solve a series of subproblems with a safety guarantee. By using the well-known convergence properties of bilevel optimization, we showed that for the single-agent case and the complete graph case, the resulting trajectories will converge to the partial minimum. Also, for the general cases with strongly connected graph, we showed the strong convergence as trust region for the consensus becomes small enough despite the lack of knowledge of the resulting converging point, which is mainly caused by the fact that only positions are the shared variables, as compared to other existing methods with first order optimality condition. Through the numerical and hardware experiments with unicycle models, we further confirmed our claims. To conclude, the distributed algorithm we presented, though lacking the optimality in general and relying on the strong assumptions on the local solver's accuracy, can provide a fast and reliable trajectory plan for distributed systems with a safety guarantee by initializing a simple path.

In the future work, we may explore how to assign agents to specific locations based on current states, and we will also develop theories and methods for time-varying graph cases where agents can determine the time-varying neighbor sets according to the current states to further reduce the communication requirements. Lastly, we are also exploring how we can implement this algorithm to other applications, such as maximum field of view coverage problems.

BIBLIOGRAPHY

[1] Giuseppe Fedele, Luigi D'Alfonso, Francesco Chiaravalloti, and Gaetano D'Aquila. Obstacles avoidance based on switching potential functions. *Journal of Intelligent Robotic Systems*, 90:387–405, 06 2018.

[2] Youmin Zhang and Hasan Mehrjerdi. A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations. pages 1087–1096, 05 2013.

[3] Mehran Mesbahi and Magnus Egerstedt. Graph theoretic methods in multiagent networks. In *Princeton Series in Applied Mathematics*, 2010.

[4] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[5] V. Gazi. Swarm aggregations using artificial potentials and sliding-mode control. *IEEE Transactions on Robotics*, 21(6):1208–1214, Dec 2005.

[6] Yuanqi Mao, Michael Szmuk, and Behcet Acikmese. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, page 3636–3641. IEEE, December 2016.

[7] Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems with state constraints. *IFAC-PapersOnLine*, 50(1):4063–4069, 2017. 20th IFAC World Congress.

[8] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behcet Acikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems, 2019.

[9] Vandenberghe L. Boyd, S. *Convex optimization*. Cambridge university press, 2004.

[10] Purnanand Elango, Dayou Luo, Abhinav G. Kamath, Samet Uzun, Taewan Kim, and Behçet Açıkmeşe. Successive convexification for trajectory optimization with continuous-time constraint satisfaction, 2024.

[11] Andrew Singletary, Karl Klingebiel, Joseph Bourne, Andrew Browning, Phil Tokumaru, and Aaron Ames. Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8129–8136, 2021.

[12] Mrdjan Jankovic, Mario Santillo, and Yan Wang. Multiagent systems with cbf-based controllers: Collision avoidance and liveness from instability. *IEEE Transactions on Control Systems Technology*, 32(2):705–712, 2024.

[13] Zhan Gao, Guang Yang, and Amanda Prorok. Online control barrier functions for decentralized multi-agent navigation. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 107–113, 2023.

[14] Pascal Bianchi and Jérémie Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

[15] Tatiana Tatarenko and Behrouz Touri. Non-convex distributed optimization. *IEEE Transactions on Automatic Control*, 62(8):3744–3757, 2017.

[16] Pascal Bianchi and Jérémie Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

[17] Stefan Vlaski and Ali H. Sayed. Distributed learning in non-convex environments—part i: Agreement at a linear rate. *IEEE Transactions on Signal Processing*, 69:1242–1256, 2021.

[18] Martin Schmidt Yasmine Beck. *A Gentle and Incomplete Introduction to Bilevel Optimization*. Optimization Online, 2023.

[19] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[20] Nicholas G. Maratos. Exact penalty function algorithms for finite dimensional and control optimization problems. 1978.

[21] DIMITRI P. BERTSEKAS. Chapter 3 - the method of multipliers for inequality constrained and nondifferentiable optimization problems. In DIMITRI P. BERTSEKAS, editor, *Constrained Optimization and Lagrange Multiplier Methods*, pages 158–178. Academic Press, 1982.

[22] Gösta Stomberg, Alexander Engelmann, and Timm Faulwasser. Decentralized nonconvex optimization via bi-level sqp and admm. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 273–278, 2022.

[23] Boris Houska, Janick Frasch, and Moritz Diehl. An augmented lagrangian based algorithm for distributed nonconvex optimization. *SIAM Journal on Optimization*, 26(2):1101–1127, 2016.

[24] Solmaz S. Kia, Bryan Van Scoy, Jorge Cortes, Randy A. Freeman, Kevin M. Lynch, and Sonia Martinez. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, 39(3):40–72, 2019.

[25] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

[26] Alexander Engelmann, Yuning Jiang, Boris Houska, and Timm Faulwasser. Decomposition of nonconvex optimization via bi-level distributed aladin. *IEEE Transactions on Control of Network Systems*, 7(4):1848–1858, 2020.

[27] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: A survey and extensions. *Mathematical Methods of Operations Research*, 66:373–407, 11 2007.

[28] Björn Geißler, Antonio Morsi, Lars Schewe, and Martin Schmidt. Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. *SIAM Journal on Optimization*, 27(3):1611–1636, 2017.

Appendix A

DERIVATION OF SVM REFORMULATION WITH CIRCULAR AGENTS

This appendix presents the derivations leading to equation (3.15).

Recall we have the Lagrangian of the form

$$\begin{aligned} &\sup_{z_k^{ij}(t)} \inf_{x^i(t)} (\sum_{l=1}^{p} \lambda_l f_l(x^i(t)) + z^T x^i(t)) + \\ &\qquad \inf_{x^j(t)} (\sum_{l=1}^{p} \mu_l g_l(x^j(t)) - z^T x^j(t)) \geq d_{min} \\ &\text{s.t. } \|z\|_2 \leq 1 \\ &\qquad \lambda \succeq 0, \mu \succeq 0 \end{aligned} \tag{A.1}$$

If we consider the circular agents with uniform radius and corresponding sets defining the interior of agents given by $S_1 := \{x^i | f(x^i) \leq 0\}, S_2 := \{x^j | g(x^j) \leq 0\}$, with $f(x^i(t)) = g(x^i(t)) = \|\frac{I}{r}x^i + b^i\|_2^2 - 1$, where $b^i = -\frac{I}{r}x_k^i(t, 0:2)$. We define $C_k^i = -rb_k^i$, the Lagrangian of the minimum distance between two agents problem becomes:

$$\begin{aligned} &L(x^i, x^j, w, \lambda^i, \lambda^j, z_k^{ij}) = \\ &\|w\|_2 + \lambda^i(\|x^i - C_k^i\| - r^2) + \lambda^j(\|x^j - C_k^j\| - r^2) + (z_k^{ij})^T(w - (x^i - x^j)) \end{aligned} \tag{A.2}$$

where $\lambda^i, \lambda^j \in \mathbb{R}^+$ are the multipliers associated with the inequality constraints, and $v \in \mathbb{R}$ is the multiplier for the equality constraints introduced by the change of variable $w = x^i - x^j$. Since the dual problem is in the form of $\sup_{\lambda^i, \lambda^j, z_k^{ij}} \inf_{x^i, x^j, w} L(x^i, x^j, w, \lambda^i, \lambda^j, z_k^{ij})$, we first need to find the infimum of the Lagrangian with primal variables. The terms associated with w is $\inf_w \|w\|_2 + (z_k^{ij})^T w$, which has optimal value of 0 when $\|z_k^{ij}\| \leq 1$. The terms for the x^i, x^j are given by $\lambda^i\|x^i - C_k^i\| - (z_k^{ij})^T x^i, \lambda^j\|x^j - C_k^j\| + (z_k^{ij})^T x^j$, which yields the optimal values of $-\frac{\|z_k^{ij}\|}{4\lambda^i} - (z_k^{ij})^T C_k^i, -\frac{\|z_k^{ij}\|}{4\lambda^i} + (z_k^{ij})^T C_k^j$, respectively. When putting all the

terms together, the dual objective becomes:

$$\begin{aligned} &g(\lambda^i, \lambda^j, z^{ij]}) = \\ &-\frac{\|z_k^{ij}\|}{4\lambda^i} - (z_k^{ij})^T C_k^i - \frac{\|z_k^{ij}\|}{4\lambda^i} + (z_k^{ij})^T C_k^j - (\lambda^i + \lambda^j) r^2 \end{aligned} \tag{A.3}$$

with some rearrangements, the final dual problem to the minimum distance between two circular agents is given by:

$$\begin{aligned} &\max_{z_k^{ij}} (z_k^{ij})^T (b_k^i - b_k^j) r - 2r\|z_k^{ij}\| \\ &\text{s.t. } \|z_k^{ij}\| \leq 1 \end{aligned} \tag{A.4}$$