2023

Resilient planning, task assignment and control for multi-robot systems against plan-deviation attacks

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

RESILIENT PLANNING, TASK ASSIGNMENT AND CONTROL FOR MULTI-ROBOT SYSTEMS AGAINST PLAN-DEVIATION ATTACKS

by

ZIQI YANG

B.S., Harbin Institute of Technology, 2015
M.Eng., Cornell University, 2017

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2023

Approved by

First Reader

Roberto Tron, PhD
Associate Professor of Mechanical Engineering
Associate Professor of Systems Engineering

Second Reader

Calin A.Belta, PhD
Professor of Mechanical Engineering
Professor of Systems Engineering
Professor of Electrical and Computer Engineering

Third Reader

Sean B. Andersson, PhD
Professor and Chair of Mechanical Engineering
Professor of Systems Engineering

Fourth Reader

Wenchao Li, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Systems Engineering

Acknowledgments

I have been a student for my entire life, and it is true that being a student is often a straightforward process with predetermined paths and established methods. However, embarking on the journey of a Ph.D. is a different experience altogether. It is the first time I have had to set my own goals, determine my own methods, and establish my own pace to reach the destination. Despite the challenges, pains, and moments of confusion throughout this process, I am proud to say that I have persevered and successfully reached the end.

In this moment of accomplishment, I am filled with immense gratitude towards my advisor, Dr. Roberto Tron. His unwavering guidance and support have played a pivotal role in my growth as a researcher. Prof. Tron not only instilled in me the importance of cultivating good research habits but also served as a constant source of inspiration. As a slow learner, I often struggled to find my own pace, but Roberto never gave up on me. His dedication and belief in my abilities served as a guiding light, illuminating my path and setting an example of hard work and perseverance.

For the collaborator of the SecuringMas team, thanks Dr. Kacper Wardega for providing the initial idea and direction of the research. His contribution laid the foundation for the work I have completed.

I am also deeply thankful to my committee members, Dr. Calin Belta, Dr. Sean Andersson, and Dr. Wenchao Li, for their invaluable feedback and suggestions. Their expertise and insights have greatly improved the quality and impact of my research.

I want to express my appreciation to the BU Robotics Lab, particularly Dr. Arman Karimian and Dr. Bee Vang, for their assistance during the early stages of my Ph.D. career. And for all the member of the robotics lab, Dawei Zhang, Dr. Zachary Serlin, Dr. Mahroo Bahreinian, Dr. Samuel Pinto, Dr. Alessio Mosca, Dr. Ye Lin, Ramón Cruz, and many others whose names I regrettably couldn't mention, I want to convey

my gratitude. It has been a truly rewarding experience working with you. I also want to thank the SE department providing me with the opportunity to pursue my Ph.D. and for their support throughout these years.

For my family, Xuejun Zhu and Jianmin Yang, thank you for your unwavering love and support. You have been my role models throughout my upbringing, and it is your dedication and achievements have inspired me to pursue the path of exploring unknown worlds. To the source of my joy, Zixin Tang, thank you for your understanding, patience, and unconditional love. Your presence in my life has been a constant source of strength and motivation.

Ziqi Yang

RESILIENT PLANNING, TASK ASSIGNMENT AND CONTROL FOR MULTI-ROBOT SYSTEMS AGAINST PLAN-DEVIATION ATTACKS

ZIQI YANG

Boston University, College of Engineering, 2023

Major Professor: Roberto Tron, PhD
Associate Professor of Mechanical Engineering
Associate Professor of Systems Engineering

ABSTRACT

The security of multi-robot systems is critical in various applications such as patrol, transportation, and search and rescue operations, where they face threats from adversaries attempting to gain control of the robots. These compromised robots are significant threats as they allow attackers to steer robots towards forbidden areas without being detected, potentially causing harm or compromising the mission. To address this problem, we propose a resilient planning, task assignment, and control framework. The proposed framework builds a multi-robot plan where robots are designed to get close enough to other robots according to a co-observation schedule, in order to mutually check for abnormal behaviors. For the first part of the thesis, we propose an optimal trajectory solver based on the alternating direction method of multipliers (ADMM) to generate multi-agent trajectories that satisfy spatio-temporal requirements introduced by the co-observation schedules. As part of the formulation, we provide a new reachability constraint to guarantee that, despite adversarial movement by the attacker, a compromised robot cannot reach forbidden areas between co-

observations without being detected.

In the second part of the thesis, to further enhance the system's performance, reliability, and robustness, we propose to deploy multiple robots on each route to form sub-teams. A new cross-trajectory co-observation scheme between sub-teams is introduced that preserves the optimal unsecured trajectories. The new planner ensures that at least one robot in each sub-team sticks to the planned trajectories, while sub-teams can constantly exchange robots during the task introducing additional co-observations that can secure originally unsecured routes. We show that the planning of cross-trajectory co-observations can be transformed into a network flow problem and solved using traditional linear program technique. In the final part of the thesis, we show that the introduction of sub-teams also improves the multi-robot system's robustness to unplanned situations, allowing servicing unplanned online events without breaking the security requirements. This is achieved by a distributed task assignment algorithm based on consensus ADMM which can handle tasks with different priorities. The assignment result and security requirements are formulated as spatio-temporal schedules and guaranteed through control barrier function (CBF) based controls.

Contents

List of Figures

List of Abbreviations

ADMM	.....	Alternating Direction Method of Multipliers
ALM	.....	Augmented Lagrangian Method
APMAPF	.....	Attack-proof Multi-Agent Path Finding
CBF	.....	Control Barrier Function
CE	.....	Central Entity
ID	.....	Independence Detection
KF	.....	Kalman Filter
MAPF	.....	Multi-Agent Path Finding
MILP	.....	Mixed-integer Linear Program
MIQP	.....	Mixed-integer Quadratic Programming
MRS	.....	Multi-Robot Systems
OD	.....	Operator Decomposition
$\mathbb{R}^2$	.....	the Real plane
RRT	.....	Rapidly-exploring Random Tree
SMT	.....	Satisfiability-Modulo-Theory

Chapter 1

Introduction

Multi-robot systems (MRS) are gaining popularity in a wide range of fields, including warehouse organization and surveillance, due to their ability to handle complex tasks through cooperation and coordination. Examples include industrial settings from Fetch and Amazon Robotics, and surveillance for forest fire monitoring and precision agriculture [Pajares, 2015, Julian et al., 2012]. However, the use of these systems also presents cyber security challenges, such as unauthorized access, malicious attacks, and data manipulation [Brunner et al., 2010]. The distributed nature of MRS, with robots communicating over a network, makes them particularly vulnerable to cyber threats. In particular, in this thesis, we are interested in scenario where an attacker over robots, and drive them to forbidden regions that would lead to confidential information exploitation, physical property damages, or even human injuries [Forrest, 2017].

Such deliberate deviations were first formulated and addressed by [Wardega et al., 2019], and named *plan-deviation attacks*. It introduced a *co-observation-based* security layer that combines path planning with the robots' physical sensors (each having the ablility to detect other robots). Co-observation requires robots to timely get close enough and *detect* abnormal behaviors (mostly deviations from pre-defined trajectories). Inspired by this research, we want to modify the planned paths for the robots so that they can observe each other frequently enough at key locations, so that compromised robots do not have the opportunity to access unauthorized areas

without being detected. We refer to these areas as *forbidden regions*, e.g., containing security-sensitive equipment, or human workers).

1.1 Overview of the proposed approach

The proposed research aims to enhance the security of MRSs against malicious takeovers and deviations by attackers. Specifically, we design an inter-robot observation plan (co-observation schedule) to ensures that, during the task period, robots are constantly in proximity to one another according to a schedule, in order to observe and detect potential hazardous behaviors. This achieved by keeping the potential region that the robots can possibly reach between each consecutive co-observations away from the forbidden regions. In this way, the plan ensure that any potential deviations to these forbidden regions will cause the corresponding robot to miss their next co-observation with other robots.

To solve situations where the planning problem is not feasible, or comes with sacrifices of task quality for the increased security, we propose to substitute each robot in the MRS with a sub-team of robots. Within each team, additional robots are sent out to travel between trajectories, providing additional co-observation opportunities across the trajectory, thus adding more freedom in optimizing trajectories while still maintaining security requirements. Our approach starts with a unsecured but optimal MRS trajectory and we formulate and solve the new co-observation plan as a multi-commodity flow problem.

To allow for deviations caused by unplanned online situations while still maintaining the security requirements in real-time, we introduce a distributed task assignment algorithm that assigns tasks with different priorities within each team to optimize the use of additional robots. The algorithm ensures that there will always be robots to fulfill the original plan and maintain the required safety protocols, so that others can

be assigned to handle online tasks.

1.2 Contributions

In the first part of the thesis, we focus on the offline planning problem including optimal MRS trajectories, reachability analysis and an alternative co-observation schedule. In Chapter 3, we propose an optimal path planning algorithm based on ADMM to handle the spatio-temporal constraints induced by the observation schedule. Additionally, our formulation allows the incorporation of a large variety of types of constraints, and the optimization of complex cost functions (e.g., uncertainty in collaborative map estimation). Our approach can only guarantee local convergence; however, this can be practically counteracted by using proper initialization techniques, such as using approximate solutions to relaxed or lower-scale versions of the problem.

In chapter 4, we incorporate the co-observation schedule introduced in [Wardega et al., 2019] into the solver, and extend the approach by analyzing the potential region a robot could possibility reach between consecutive co-observations. By enforcing an empty intersection between the forbidden regions and the approximate reachability regions, we can guarantee that if an attacker were to take control of the robots, they would not be able to perform an undetected attack (i.e., entering forbidden regions while still able to meet observation schedules at the same time). The reachability constraint is formulated using an ellipsoidal bound of the reachability region. As a secondary contribution, we introduce the concept of *Householder rotations* for defining rigid body changes of coordinates that are differentiable.

In Chapter 5, we propose to incorporate redundancy in the form of *sub-team*s of multiple robots in place of individual robots along each route. This allows robots to deviate from their assigned sub-team, and join others to perform cross-trajectory co-observations, thereby securing multiple trajectories. We propose a formulation of

the multi-flow problem on unsecured MRS trajectories to plan the cross-trajectory co-observations that can preserve the security against plan-deviation attacks.

In the second part of this thesis, we shift our focus to real-time online control problem of MRS. In Chapter 6, we provide a way to dispatch and compute deviation and regroup times for sub-teams using an approximated reachability analysis from Chapter 4. A computationally inexpensive distributed task assignment algorithm, which utilizes Alternating Direction Method of Multipliers (ADMM) and consensus-like algorithms, is introduced to handle different priority constraints, such as guaranteeing preplanned tasks versus online tasks. We further combine the assignment algorithm with Control Barrier Functions (CBFs) for real-time control, demonstrating its applicability for security in multi-robot systems.

Chapter 2

Background and Preliminaries

2.1 Multi-robot system

Our focus is on a centralized multi-robot system (MRS) model, which includes a set of robots and a central entity (CE) responsible for communication and the management of the robots. The CE computes multi-robot motion plans, denoted as q, that enable the robots to carry out the required tasks while ensuring adherence to safety constraints. In our work, the safety constraint manifest themselves as *forbidden regions*, i.e. locations in the environment that are inaccessible to the robots due to various factors such as human operator workspaces, hazardous or fragile equipment storage, or sensitive and confidential information locations.

2.2 Attacker model and plan-deviation attacks

In this work, we consider *plan-deviation attacks* as introduced by [Wardega et al., 2019] to model potential threats to multi-agent systems. Specifically, we assume that a robot in the system has been compromised by an attacker who intends to violate the safety constraints by entering forbidden areas undetected. The attacker has full knowledge of the motion plan, and aims to masquerade the compromised robot as a legitimate one to the CE. The attacker intends to let the compromised robot perform deviations from the nominal plan and seek access to forbidden areas. We refer to these malicious deviations as plan-deviation attacks. An undetected plan-deviation

occurs when a compromised robot deviates from the motion plan while providing a false self-report to the central entity (CE) about its location. Under our model, we consider such deviations to go undetected by the CE as long as the self-reports of all other robots remain unchanged.

2.3 Optimal trajectory planning problem

The planning against plan-deviation attacks, in our instance, is an optimal trajectory optimization problem for multiple agents to minimize arbitrary smooth objective functions while satisfying different types of spatio-temporal constraints. We denote as $q_{ij} \in \mathbb{R}^m$ the position of agent i at the discrete-time index j, with m representing the dimension of the state space. For a team of n agents, and a task time horizon of T, the overall trajectory of the multi-agent system can be represented as an aggregated vector $\mathbf{q} \in \mathbb{R}^{nmT}$. The goal of our path planning problem is to minimize or maximize an objective function $\Phi(\mathbf{q})$ under a set of nonlinear constraints described by a set Ω, formally:

$$\begin{aligned} &\min / \max \quad \Phi(\mathbf{q}) \\ &\text{subject to} \quad \mathbf{q} \in \Omega. \end{aligned} \tag{2.1}$$

2.4 Minimum uncertainty mapping

In this work, as an example application of this formulation, we assume that there exists an underlying slowly-time-varying scalar or vector field $\mathbf{x}$ whose value needs to be estimated at a given number of discrete locations. We then define the cost function $\Phi(\mathbf{q})$ to be an approximation of the uncertainty at any of the given points, and we aim to minimize such maximium uncertainty. We focus on a particular choice of objective function $\Phi(\mathbf{q})$, in the context of a mapping and estimation applications, for the optimization problem (2.1). This mapping application is used throughout the

thesis as a model application to test all the algorithms.

To explore unknown spatially-distributed fields, autonomous agents are usually required to traverse the entire unknown environment while collecting sensory data to estimate a corresponding map. We model the map as a set of locations of interests arranged on a regular grid, where at each location, we assume that there is a corresponding slowly-time-varying quantity. For our purposes, we are interested in finding paths that best reconstruct the field, i.e., that achieve the minimum uncertainty.

We formulate the estimation problem using the information form of Kalman Filters (KFs, [Anderson and Moore, 2012]), which provides a straightforward way to quantify the uncertainty in the map model. In particular, the information form makes it more convenient to model states and measurements as having zero information (i.e., infinite variance). Since the uncertainty of the estimates for KFs does not depend on the actual measurements for path planning purposes, we can ignore the estimates of each state, and instead focus on the information matrix Y which is the inverse of the estimated covariance. The information matrix can then be initialized as zero ($Y_0 = 0$) to model the fact that we do not have any a priori information on the field. The information matrix update step can be written as:

$$Y_j = (I + Y_{j-1}Q_{j-1})^{-1} Y_{j-1} + I_j(q_j) \tag{2.2}$$

$$I_j(q_j) = R_j(q_j)^{-1} \tag{2.3}$$

where $I_j \in \mathbb{R}^{nm}$ is the inverse of the measurement covariance matrix which is a function of q_j. For example, we can formulated (2.3) using a Gaussian radial basis function, with the lth element in I be presented as:

$$[I_j]_l = \sum_{i=1}^{n} K \exp\left(-\frac{\|c_l - q_{ij}\|^2}{2\sigma_c^2}\right), \quad l \in [1, \ldots, n \times m] \tag{2.4}$$

where c_l is the coordinate of the field restoration location that needs to be calculated,

and σ_c depends on the radius and accuracy of the robot's sensor.

We then formulate the map exploration problem with the goal of finding the trajectory $\mathbf{q}$ that maximizes the minimum information along any of the directions of Y_T, with T being the task time horizon. The optimization problem in (2.1) becomes:

$$\begin{aligned} \underset{\mathbf{q}}{\text{argmax}} \quad & \text{softmin}\left(\text{diag}\left(Y_T(\mathbf{q})\right)\right) \\ s.t. \quad & \forall i \in [0, T], \forall j \in N, \\ & \mathbf{q} \in \Omega, \end{aligned} \tag{2.5}$$

where the function softmin is defined as a smooth approximation of the min function, defined as:

$$\text{softmin}(a_1, \ldots, a_n) = \log(e^{a_1} + \cdots + e^{a_n}). \tag{2.6}$$

As required by the trajectory optimization objective function, the multi-robot systems are expected to plan trajectories that efficiently cover the entire task space, often resembling a boustrophedon-like pattern.

2.5 Related works

2.5.1 Multi-agent path planning

Traditional multi-agent path planning problems are usually divided in two ways: centralized approaches and distributed approaches. Centralized approaches treat the multi-agent as a single agent system which can be solved using a single agent planning method: graph search based algorithms like Dijkstra Algorithms, A^* [Choset et al., 2005, Dolgov et al., 2010]; sampling based algorithms like $RRT, RRT^*, RRT^\#$ [LaValle, 2006, Karaman et al., , Hauer and Tsiotras, 2017]; and optimal solvers (which will be discussed later). This approach can obtain the optimal results but scales poorly. Distributed approaches treat each agent separately. By splitting the higher dimensional problem into several lower dimensional ones, and solving individually, distributed approaches significantly reduce the computational cost. But, in general, there is no

guarantee of optimality and completeness.

Many recent works have focused on a combination of the two approaches. Biased Cost Pathfinding [Geramifard et al., 2006] focuses on collision prevention by using repulsive potential fields at the collision locations during the planning phase. In [Standley, 2010], the authors present an algorithm based on operator decomposition (OD) and Independence Detection (ID) technique. Agents are first decoupled into non-independent subgroups through ID, then OD computes and update these subgroups in an arbitrary but fixed order.

An area of high interest and activity is optimization based approaches. These approaches are customizable to the task's specific needs (e.g. maximum surveillance coverage, minimal energy cost) and constraints (e.g. speed limit and avoid obstacles). While many motion planning tasks require a non-convex constraint problem formulation, most contributors focused on convex problems, and only allow for a few types of pre-specified non-convex constraints through convexification [Liu and Lu, 2014] [Van Parys and Pipeleers, 2016] [Schulman et al., 2014]. Several optimization techniques like MIQP [Mellinger et al., 2012] and ADMM [Bento et al., 2013] have been used to reduce computational complexity, and to incorporate more complex non-convex constraints.

Through there exist some rich literature on multi-agent pathfinding (MAPF) problems [Stern et al., 2019], only a few have taken safety requirements into consideration. [Zhu and Martinez, 2013] focus on physical layer resilient control, models the game for CPS security, and provides resilience through a game-theoretic approach. [Gil et al., 2017] made use of the physics of wireless signals, information from sensing infrastructure and proximity graph of vehicles, leverage the physics of environment to defend attacks. Our paper took a similar approach of using physical layers to detect and mitigate attacks.

Regarding the specific application considered in this thesis, there exists some rich literature on multi-agent mapping and exploration problems. The goal of these works is to maximize the coverage of information of an unknown environment [Julian et al., 2012, Schwager et al., 2011] by utilizing the information exchanged inside the network to solve the problem in a distributed fashion. For example, Rapidly-exploring Random Cycles (RRC) [Lan and Schwager, 2016], a variant of RRT, is introduced to generate a periodic trajectory for multiple sensing robots to explore a dynamic spatio-temporal field.

2.5.2 Multi-agent task assignment

As discussed in the introduction, in this paper we propose to use sub-teams to deal with online events. This requires the system to optimally assign tasks according to predefined priorities (i.e. pre-planned tasks and online tasks) [Khamis et al.,]. Traditionally, multi-agent task assignment problems have been addressed using either distributed or centralized approaches. With centralized approaches, the individual agent's information is communicated to a central agent or server, which then generates a plan for the entire system. By taking into account the capabilities and limitations of all agents and tasks, the central planner is able to optimize the overall system performance [Coltin and Veloso, 2010, Liu and Kroll, 2012, Jin et al., 2003]. However, the centralized method sacrifices robustness and scalability, exposing the system to risks of failures at the central entity and limiting the range to where centralized communications are possible.

Decentralized approaches, on the other hand, rely on local communications between agents, and offer resilience to a single point of failure, as well as scalability in team size and mission range. Common methods include consensus-based algorithms and auction algorithms [Quinton et al., 2023, Cao et al., 2012]. Consensus usually relies on local communication to converge to a common value. To improve convergence

time, some research suggests aiming for an approximate consensus [Alighanbari and How, 2005, Dionne and Rabbath, 2007]. However, these methods can't guarantee a conflict-free solution. Auction algorithms guarantee a conflict-free solution, but are not robust to distributed network topologies, and cannot effectively deal with multiple tasks [Sariel and Balch, 2005, Walsh and Wellman, 1998]. Some approaches provide improvements for the latter case, e.g., by running sequential auctions [Sariel and Balch, 2005, Sujit and Beard, 2007], or incorporating consensus methods [Choi et al., 2009]. In any case, however, it is not clear that existing consensus- or auction-based methods can be applied to our application, which requires assignments with different priorities to ensure security requirements.

Chapter 3

Optimal Multi-Agent Path Planning using Alternating Direction Method of Multipliers

In this chapter, we proposed a flexible formulation of problem (2.1) that can generate optimal trajectories for multi-agent systems while satisfying traditional and new path planning constraints.

3.1 Optimal multi-robot path planning with spatio-temporal constraints

The APMAPF algorithm proposed in [Wardega et al., 2019] provide a solution against plan-deviation attacks in a restricted setting, where the robots can only move on a subset of a four-connected grid, and the only type of cost that can be (indirectly) optimized is the maximum path length. The proposed algorithm is based on a Satisfiability-Modulo-Theory (SMT) solver, for which the complexity scales (in a worst-case scenario) exponentially with the problem size. To extend this planner to a continuous configuration space, and to allow for more general tasks that can be formulated as arbitrary smooth cost functions, we introduce a trajectory optimization problem for multiple agents, and build a solver to deal with the spatio-temporal constraints introduced by the APMAPF solutions.

This proposed solver is based on the Alternating Direction Method of Multipliers

(ADMM), a variation of the Augmented Lagrangian Method (ALM, [Boyd et al., 2011]). We choose ADMM for its empirically demonstrated ability to deal with non-convex and non-smooth optimization problems [Wang et al., 2019]. To the best of our knowledge, our path planning algorithm is the only one flexible enough to handle the spatio-temporal constraints that we will introduce in Chapter 4.

3.2 Preliminaries

3.2.1 Alternating Direction Method of Multipliers

The basic idea behind ADMM [Boyd et al., 2011] is to split the constraints from the objective function using a different set of variables $\mathbf{z}$, and then solve an Augmented Lagrangian formulation of the optimization problem in (2.1). More specifically, we can rewrite the constraint $\mathbf{q} \in \Omega$ using an indicator function Θ, and include it in the objective function. Then problem (2.1) can be rewritten as:

$$\begin{aligned} \min \quad & \Phi(\mathbf{q}) + \Theta(\mathbf{z}) \\ s.t. \quad & \mathbf{q} - \mathbf{z} = 0 \end{aligned} \tag{3.1}$$

where Θ is the indicator function of Ω. The corresponding augmented Lagrangian can then be formulated as:

$$L_\rho(\mathbf{q}, \mathbf{z}, \mathbf{u}) = \Phi(\mathbf{q}) + \Theta(\mathbf{z}) + (\rho/2)\|\mathbf{q} - \mathbf{z} + \mathbf{u}\|_2^2, \tag{3.2}$$

In the traditional application of ADMM, the duplicated variables $\mathbf{z}$ are solved through a projection to the constraint set Ω. However, in path planning problems, some constraints are non-convex, rendering the projection step more difficult (due to the presence of multiple local minima). To deal with this problem, we propose a minor generalization of the ADMM formulation (3.1) where we allow $\mathbf{z}$ to replicate an

arbitrary function of the main variables $\mathbf{q}$ (instead of being an exact copy):

$$\begin{aligned} \max \quad & \Phi(\mathbf{q}) + \Theta(\mathbf{z}) \\ s.t. \quad & D(\mathbf{q}) - \mathbf{z} = 0 \end{aligned} \tag{3.3}$$

where $D(\mathbf{q}) = [D_1(\mathbf{q})^T, \ldots, D_l(\mathbf{q})^T]^T$ is a vertical concatenation of different functions for different constraints.

The update steps of the ADMM algorithm can be then denoted as:

$$\mathbf{q}^{k+1} := \underset{\mathbf{q}}{\operatorname{argmin}}(\Phi(\mathbf{q}^k) + \frac{\rho}{2}\|D(\mathbf{q}) - \mathbf{z}^k + \mathbf{u}^k\|_2^2) \tag{3.4a}$$

$$\mathbf{z}^{k+1} := \Pi_\zeta(D(\mathbf{q}^{k+1}) + \mathbf{u}^k) \tag{3.4b}$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}, \tag{3.4c}$$

where Π_ζ is the new projection to the modified constraint set ζ, $\mathbf{u}$ represents a scaled dual variable that intuitively accumulates the sum of primal residuals

$$\mathbf{r}^k = D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}. \tag{3.5}$$

Checking the primal residuals alongside with the dual residuals

$$\mathbf{s}^k = -\rho(z^k - z^{k-1}), \tag{3.6}$$

after each iteration, the steps are reiterated until convergence when the primal and dual residuals are small, or divergence when primal and dual residual remains large after a fixed large number of iterations.

Since $D(\mathbf{q})$ is the vertical concatenation of $D_l(\mathbf{q})$, the projection of each set ζ_l is independent for each constraint and can be computed separately. The advantage of this formulation is that we can choose $D(\mathbf{q})$ such that the new constraint set ζ becomes simple to compute; however, the drawback is that we move the non-convexity to the

primal cost function, i.e., in the update for $\mathbf{q}$ in (3.4a). Additionally, the function $D(\mathbf{q})$ can be used to select only the subset of the variables on which a constraint depends.

3.2.2 Adaptive penalty parameter

In our application, a static penalty parameter might cause the result to get stuck in local minimal. Thus, we would like to have the first iterations of the algorithms focus on obtaining a (possibly global) optimal initial guess (with less regard for the constraints), while gradually bringing back the constraints during the optimization, modifying the initial guess to eventually satisfy all the constraints following the standard practice. The penalty parameter is updated according to:

$$\rho^{k+1} = \begin{cases} \tau^{incr}\rho^k & \text{if } \|\mathbf{r}^k\|_2 \leq \mu\|\mathbf{s}^k\|_2, \\ \rho^k/\tau^{decr} & \text{if } \|\mathbf{s}^k\|_2 \leq \mu\|\mathbf{r}^k\|_2, \\ \rho^k & \text{otherwise.} \end{cases} \tag{3.7}$$

where $\mathbf{r}^k$ is the primal residual given in (3.5), μ, $\tau^{incr} > 1$ and $\tau^{decr} > 1$ are adjustable constants used to adjust the penalty (in our case $\mu = 2$, $\tau^{incr} = \tau^{decr} = 1.4$), and $\mathbf{s}^k$ is the dual residual:

$$\mathbf{s}^k = -\rho\left(\mathbf{z}^k - \mathbf{z}^{k-1}\right). \tag{3.8}$$

3.2.3 Complete algorithm

Algorithm 1 is used to solve the generalized problem in (3.3) with dynamic penalty updates. For convergence proofs regarding this algorithm, see [Boyd et al., 2011].

3.3 Path planning constraints in ADMM based solver

In this section, we provide mathematical descriptions for a variety of constraints. As introduced in Section 3.2.1, constraints are decoupled and reformulated as mapping

Algorithm 1 ADMM iteration for problem 3.3

Initialize $\mathbf{q}^0, \mathbf{z}^0, \mathbf{u}^0$
while $r^k \neq 0$, $s^k \neq 0$ **do**
  $\mathbf{q}^{k+1} := \text{argmin}_{\mathbf{q}}(\Phi(\mathbf{q}^k) + \frac{\rho^k}{2}\|D(\mathbf{q}) - \mathbf{z}^k + \mathbf{u}^k\|_2^2)$
  $\mathbf{z}^{k+1} := \Pi_{\Omega}(D(\mathbf{q}^{k+1}) + \mathbf{u}^k)$
  $\mathbf{u}^{k+1} := \mathbf{u}^k + D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}$
  penalty update (3.7)
end while

functions $D(\mathbf{q}) = \mathbf{z}$ and constrained sets ζ.

3.3.1 Path Planning Constraints

We consider the following types of traditional path planning constraints:

1) Start and end locations: We assume that the start and end locations for each agent, i.e., q_{i0} and q_{iT}, are given.

2) Velocity constraints: We enforce approximate constraints on the dynamics of the agents by assuming that they can move a maximum travel distance in any arbitrary direction over discrete time periods. This constraint could be easily modified to use more detailed models.

3) Convex obstacle constraints: We model areas that cannot be entered by the agents with convex polygons. These areas can represent physical obstacles, or forbidden regions that the agent should not access (e.g., because they contain sensitive information). If the environment contains non-convex obstacles, they can be still modeled using the union of (possibly overlapping) convex obstacles.

4) Waypoints with flexible deadlines: We assume that we are given locations that need to be visited by a given agent in a given time window, although the precise instant in that time window can be chosen by the planner.

In Chapter 4, we introduce additional security constraints:

1) Co-observation schedules constraints: We assume that two agents can detect each other's presence when they are at a distance of at most d_{max} from each other at desired time t and near location q'.

2) Reachability constraints: We model the robots' reachability areas between every consecutive co-observations as ellipsoids and enforce a empty intersection between the ellipsoids and forbidden regions.

3.3.2 Start and end locations

For each agent, we fix the starting and end locations, i.e., q_{i0} and q_{iT}. These are simply enforced by removing them from the optimization variables. For other applications, it is possible to include q_{iT} as a variable, and add the corresponding linear constraint to the optimization.

3.3.3 Velocity constraint

Since we are using a discrete formulation, the velocity of an agent can be approximated by taking the difference between adjacent waypoints on the trajectory. We then define the function $D(\mathbf{q})$ to return the velocity vectors for the i-th agent at time step j:

$$D_{ij}(\mathbf{q}) = q_{ij} - q_{i(j-1)}, \quad j \in \{1, \ldots, T\} \tag{3.9}$$

and the constraint set is:

$$\zeta_{ij} = \big\{ z \mid \|z\| \leq v_{max} \big\}. \tag{3.10}$$

The projection operator $\Pi_\zeta(z)$ for this constraint implies that the projection of the vector z is inside a sphere with a radius of v_{max} which can then be written as:

$$\Pi_\zeta(z) = \begin{cases} v_{max} \frac{z}{\|z\|} & \text{if } \|z\| > v_{max}, \\ z & otherwise. \end{cases} \tag{3.11}$$

3.3.4 Convex obstacles

In our motivating scenarios, there are areas that the agents should not visit, because they represent forbidden regions or obstacles to avoid. We model these zones using convex polygons defined by several hyperplanes and having the normal vectors of these hyperplanes pointing outside the obstacles. We enforce the constraints at each discrete time step q_{ij} (enforcement of the constraints between these points can be achieved by making the obstacles sightly bigger than their actual size). For a convex area, waypoints stays unchanged if they are outside the region, if they are inside the region, the constraint function $D(\mathbf{q})$ returns the least negative distance to each hyperplanes that defines the boundary of the region, represented as:

$$D_i(\mathbf{q}) = \left[d_{11} \dots d_{nm}\right]^{\mathrm{T}}, \tag{3.12}$$

where

$$d_{ij} = \max(\min(d_{ijk}, 0)) \tag{3.13}$$

$$d_{ijk} = p_{ij}^{\mathrm{T}} n_k - m_k, \tag{3.14}$$

and n_k is the normal vector and m_k is the scalar offset defining the k-th hyperplane. The corresponding constraint set is simply

$$\zeta = \{z \mid z = 0\}, \tag{3.15}$$

with a projection function:

$$\Pi_\zeta(z) = 0. \tag{3.16}$$

The motivating idea for (3.12) and (3.16) is to find waypoints inside the zone and project them to the closest boundary. As mentioned before, non-convex obstacles can be handled by the union of (possibly overlapping) convex obstacles.

3.3.5 Waypoints with flexible deadlines

In this type of constraint, we assume that an agent needs to go through a spherical ball around a given point p with a radius of d_{max} at some time instant j belonging to a given time window $[t_1, t_2]$. In this case, we define the function $D(\mathbf{q})$ to return the smallest distance from the point p to any point on the trajectory restricted to the relevant time window. This can be expressed precisely in the following form:

$$D_i(\mathbf{q}) = \min_{i\in\{1\ldots,n\}, j\in\{t_1,\ldots,t_2-1\}} \big(\mathrm{dist}(p, \overrightarrow{q_{ij}q_{i(j+1)}})\big), \tag{3.17}$$

with the constraint set:

$$\zeta = \{z \mid z < z_{max}\}, \tag{3.18}$$

where $\mathrm{dist}(p, \overrightarrow{q_{ij}q_{i(j+1)}})$ returns the distance between the fixed point p and the segment $\overrightarrow{q_{ij}q_{i(j+1)}}$. Note that this function returns the smallest distance between (p, q_{ij}) and $(p, q_{i(j+1)})$ if the projection of the point p does not lie on the line segment $\overrightarrow{q_{ij}q_{i(j+1)}}$; as a consequence, this constraint does not need to be satisfied exactly at one of the points on the discretized trajectory, but it can also be satisfied "en route" on the segment between them. The projection $\Pi_\zeta(z)$ for this constraint can be written as:

$$\Pi_\zeta(z) = \min(z, z_{max}). \tag{3.19}$$

Note that (3.18) and (3.19) are equivalent to (3.10) and (3.11), except for the fact that the quantity in (3.17) is always a positive scalar.

3.4 Result and simulation

In this section, we apply our ADMM path planning algorithm to an instance of a map exploration problem, and test the algorithm in both simulations and an experimental testbed. The environment to be explored, with three agents, is a $10m \times 10m$ region.

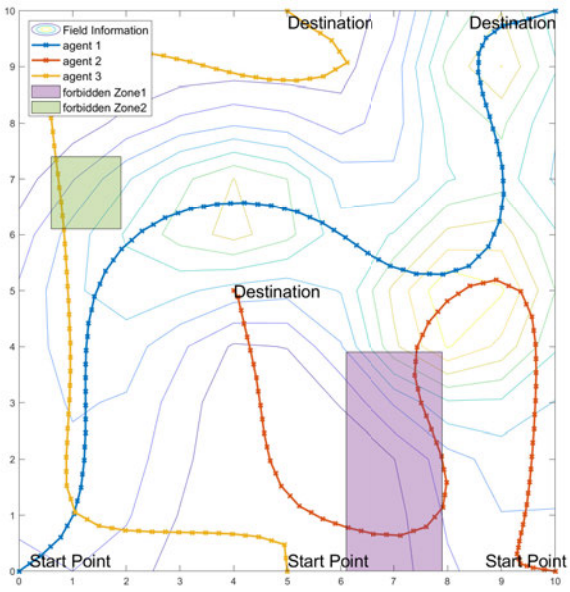


Figure 3·1: Baseline test with only velocity constraint, robots' start locations and destinations are fixed, forbidden regions and obstacles are not considered. The contour in the back shows the resulting exploration quality.

We set the sensor accuracy for each agent in (2.4) to $\sigma_c = 1$; we assume that the robots can meaningfully collect information on the underlying vector field for data locations no further than $3m$. The maximum velocity constraint v_{max} is set to $0.5m/dt$.

Our baseline test is one that only considers the velocity constraint; the result can be seen in Figure 3·1. The contours show the estimated accuracy for the map which is the result of the information matrix (the larger the better). The elements in the resulting entries of the information matrix for this test has a maximum of 5.5 and a minimum of 1.8. Note that the forbidden regions (obstacles) were not included as constraints, which explains the violations by the agents. The trajectories are initialized as straight lines with the agents moving at uniform speed from their start to their goal locations. After running our proposed method, the trajectories assume boustrophedon-like patterns. These patterns have been traditionally proposed

for coverage problems, but here they appear as a byproduct of our cost function while taking into account the spatio-temporal behavior of the uncertainty; for instance, the trajectories of agent 1 and agent 2 appear to be very close, which would be suboptimal in terms of minimizing the maximum uncertainty; in fact, agent 1 reaches the point closest to agent 2's path well after agent 2 has left the corresponding point on its trajectory; in a sense, agent 1 and agent 2 automatically take turn to cover that location.

The solution of the baseline test in Figure 3·2 is used to initialize another test with some additional obstacles (shown as rectangles in Figure 3·1). Note that the initial trajectories, from the baseline test, is infeasible since the agents cross the forbidden areas (see Figure 3·1); our algorithm is capable of handling the new constraints and returning a feasible solution, as shown in Figure 3·2. Note that the algorithm corrects the trajectory for agent 2 from near the right edge of the purple obstacle to the left of the obstacle. Resulting entries of the information matrix for this case have a max of 7.2 and a min of 1.4.

3.5 Summary

In this chapter, we have presented a path planning algorithm that can generate optimal trajectories for multi-agent systems while satisfying complex spatio-temporal constraints. This planner can be used to support new methods to enhance the security of networks of robots against malicious take-overs of agents. Our solution is based on an application and modification of the ADMM framework. Simulation and experimental results show that our method, while based on local optimization, is able to obtain solutions for non-trivial planning problems involving cumulative cost functions and waypoints with flexible deadlines. This planner provide a method for generating plans with flexible choices of objectives and constraints. However, the computation is not

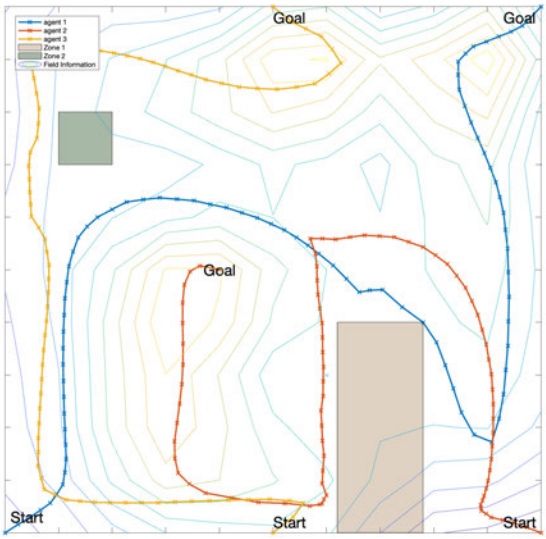


Figure 3·2: Forbidden region and obstacle are added to the baseline test.

fast enough for online computation, making it only appropriate as a offline planner. In the next few chapters, we incorporated additional security constraints in chapter 4, and generating online control algorithms considering the security of the whole system in chapter 6.

Chapter 4

Co-observation Schedule and Reachability Constraints

In this chapter, we provide methods transforming co-observation schedules from the APMAPF algorithm into constraints of the solver introduced in Chapter 3. We also provide the definition of an *ellipsoidal reachability region*, and provide a differentiable map to transform such region in a canonical axis-aligned form where the operator Π_ζ and its differential can be obtained; these operators are then extended to the general case via the aforementioned transform, and then used in our ADMM-based solver.

4.1 Co-observation schedule and reachability region

Besides traditional path planning constraints, additional security constraints need to be incorporated into the path planning problem to prevent cases where an attacker changes a robot's path between two observations to reach a forbidden region without being detected. The work of [Wardega et al., 2019] provide exact solution for grid-world configuration. Using the onboard sensing capabilities, robots can perform inter-robot observations as an additional security measure. Each robot' self-report includes not only its own status, but also additional observations of other robots' status. For example, robot i's self-report may include its observation of robot j, "i report its arrival at v and i observes j at location w". Similarly, robot j also reports its observations of robot i to the CE. This solution provides paths with an observation schedule such that any attempts by a compromised robot to violate the

safety constraints would necessarily break the observation plan and be detected.

When transforming the solution to continuous configuration spaces, additional analysis is needed on the reachability of robots in order for the security of co-observation schedule to hold. More broadly, the problem of solving a path optimization problem with constraints based on the sets of locations that the agents could *potentially* reach, which we call *reachability regions*, has not received attention in the literature. This constraint is formulated using an ellipsoidal bound of the reachability region, and the idea of using an ellipsoidal bound to limit the search space is inspired by the *heuristic sampling domain* introduced by [Gammell et al., 2014] in the context of the `RRT`* path planning algorithm.

We formulate a way to enforce an empty intersection between forbidden regions and ellipsoidal reachability region while optimizing the trajectory, such that if an attacker takes control of the robots, they cannot perform an undetected attack without breaking the co-observation schedule. We propose a mathematical formulation of reachability regions that is compatible with the solver from Chapter 3 as a spatio-temporal constraint. As a secondary contribution, we introduce the concept of *Householder rotations* for defining differentiable rigid-body changes of coordinates.

4.2 Preliminaries

In this section, we review various mathematical concepts that will provide the foundations and context for our novel constraints.

4.2.1 Differentials

We define the differential of a map $f(x) : \mathbb{R}^m \to \mathbb{R}^n$ at a point x_0 as the unique matrix $\partial_x f \in \mathbb{R}^{n \times m}$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t} f\big(x(t)\big)\bigg|_{t=0} = \partial_x f\big(x(0)\big)\dot{x}(0) \tag{4.1}$$

where $t \mapsto x(t) \in \mathbb{R}^n$ is a smooth parametric curve such that $x(0) = x$ with any arbitrary tangent $\dot{x}(0)$. For later part of this paper, we will use $\dot{f}$ for $\frac{\mathrm{d}}{\mathrm{d}t}f$ and $\partial_x f$ for $\frac{\partial f}{\partial x}$. The differentials $\partial_x f$ is derived through (4.1) having $\dot{f}$ divided by $\dot{x}$.

With a slight abuse of notation, we use the same notation $\partial_x f$ for the differential of a matrix-valued function with scalar arguments $f : \mathbb{R}^R \to \mathbb{R}^{m \times n}$ Note that in this case (4.1) is still formally correct, although semantically different.

4.2.2 Householder rotations

To formulate the operator Π_ζ for our novel constraint, we define a differentiable transformation of the constraint set to a canonical form. This transformation includes a rotation that we derive from a modified version of Householder transformations [Householder, 1958]. With respect to the standard definition, our modification ensures that the final operator is a proper rotation (i.e., not a reflection). We call our version of the operator a *Householder rotation*. In this section we derive Householder rotations and their differentials for the 3-D case; the 2-D case can be easily obtained by embedding it in the $z = 0$ plane.

Definition 1. *Let $\nu_\mathcal{F}$ and $\nu_\mathcal{E}$ be two unitary vectors ($\|\nu_\mathcal{F}\| = \|\nu_\mathcal{E}\| = 1$). Define the normalized vector u as*

$$u' = \nu_\mathcal{F} + \nu_\mathcal{E}, \tag{4.2}$$

$$u = \frac{u'}{\|u'\|}. \tag{4.3}$$

The Householder rotation *$H(\nu_\mathcal{F}, \nu_\mathcal{E})$ is defined as*

$$H(\nu_\mathcal{F}, \nu_\mathcal{E}) = 2uu^{\mathrm{T}} - I. \tag{4.4}$$

The main property of interest for our application is the fact that H is a rotation mapping $\nu_\mathcal{F}$ to $\nu_\mathcal{E}$, as shown by the following.

Proposition 1. *The matrix H has the following properties:*

1) It is a rotation, i.e.

(a) $H^{\mathrm{T}}H = I$;

(b) $\det(H) = 1$.

2) $\nu_{\mathcal{E}} = H\nu_{\mathcal{F}}$.

Proof. For subclaim *1)*a:

$$H^{\mathrm{T}}H = H^2 = 4uu^{\mathrm{T}}uu^{\mathrm{T}} - 4uu^{\mathrm{T}} + I^2 = I, \tag{4.5}$$

since $u^{\mathrm{T}}u = 1$.

For subclaim *1)*b, let $U = [u \quad u_1^{\perp}, u_2^{\perp}]$, where $u_1^{\perp}, u_2^{\perp}$ are two orthonormal vectors such that $I = UU^{\mathrm{T}} = uu^{\mathrm{T}} + u_1^{\perp}(u_1^{\perp})^{\mathrm{T}} + u_2^{\perp}(u_2^{\perp})^{\mathrm{T}}$; then, substituting I in (4.4), we have that the eigenvalue decomposition of H is given by

$$H = U\,\mathrm{diag}(1, -1, -1)U^{\mathrm{T}}. \tag{4.6}$$

Since the determinant of a matrix is equal to the product of the eigenvalues, $\det(H) = 1$.

For subclaim *2)*, first note that $Hu = 2uu^{\mathrm{T}}u - u = u$. It follows that the sum of $\nu_{\mathcal{F}}$ and $\nu_{\mathcal{E}}$ is invariant under H:

$$H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) = Hu\|\nu_{\mathcal{F}} + \nu_{\mathcal{E}}\| = u\|\nu_{\mathcal{F}} + \nu_{\mathcal{E}}\| = \nu_{\mathcal{F}} + \nu_{\mathcal{E}}, \tag{4.7}$$

and that their difference is flipped under H:

$$H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) = 2uu^{\mathrm{T}}(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) - (\nu_{\mathcal{F}} - \nu_{\mathcal{E}})^2 = -(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}). \tag{4.8}$$

Combining (4.7) and (4.8) we obtain

$$H\nu_{\mathcal{F}} = \frac{1}{2}\big(H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) + H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}})\big) = \nu_{\mathcal{E}} \tag{4.9}$$

□

We compute the differential of H implicitly using the relation (4.1). We will use the notation $[v]_{\times} \colon \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ to denote the matrix representation of the cross product with the vector v, i.e.,

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{4.10}$$

such that $[v]_\times w = v \times w$ for any $w \in \mathbb{R}^3$. One can verify by direct computation the following property:

$$wv^{\mathrm{T}} - vw^{\mathrm{T}} = [[v]_\times w]_\times. \tag{4.11}$$

Proposition 2. *Let $\nu_\mathcal{F}$ represent a parametric curve. The derivative of H with respect to $\nu_\mathcal{F}$ can be written as:*

$$\dot{H} = H[-2M\dot{\nu}_\mathcal{F}]_\times \quad , \tag{4.12}$$

where the matrix $M \in \mathbb{R}^{3\times 3}$ is given by

$$M = [u]_\times \frac{\left(I - uu^{\mathrm{T}}\right)\left(I - \nu_\mathcal{F}\nu_\mathcal{F}^{\mathrm{T}}\right)}{\|u'\|}. \tag{4.13}$$

Proof. From the definition of H in (4.4), we have

$$\dot{H} = 2(\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) \tag{4.14}$$

Recall that $\dot{u} = \frac{1}{\|u'\|}(I - uu^{\mathrm{T}})\dot{u}'$ (see, for instance, [Tron and Daniilidis, 2014]), which implies $(I - uu^{\mathrm{T}})\dot{u}' = \dot{u}'$. It follows that $\dot{u}$ flips sign under the action of H^{T}:

$$\begin{aligned} H^{\mathrm{T}}\dot{u} = (2uu^{\mathrm{T}} - I)\frac{\left(I - uu^{\mathrm{T}}\right)}{\|u'\|}\dot{u}' & \\ = \frac{1}{\|u'\|}(2uu^{\mathrm{T}} - I - 2uu^{\mathrm{T}}uu^{\mathrm{T}} + uu^{\mathrm{T}})\dot{u}' & \\ = -\frac{1}{\|u'\|}(I - uu^{\mathrm{T}})\dot{u}' = -\dot{u} & \end{aligned} \tag{4.15}$$

Inserting $HH^{\mathrm{T}} = I$ in (4.14), and using 4.11, we finally have

$$\begin{aligned} \dot{H} =& 2HH^{\mathrm{T}}(\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) = 2H(-\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) \\ =& -2H[[u]_\times \dot{u}]_\times \\ =& -2H\left[[u]_\times \frac{\left(I - uu^{\mathrm{T}}\right)\left(I - \nu_\mathcal{F}\nu_\mathcal{F}^{\mathrm{T}}\right)}{\|u'\|\,\|\nu_\mathcal{F}\|}\dot{\nu}_\mathcal{F}\right]_\times \\ =& -2H[M\dot{\nu}_\mathcal{F}]_\times, \end{aligned} \tag{4.16}$$

which is equivalent to the claim. □

4.3 Co-observation constraint

The result of APMAPF provide us with a grid world solution trajectory alongside with a co-observation schedule requiring robots to meet with another one according to the defined schedule. Thus, as introduced in 3.3.1, the co-observation constraints is used to guarantee that, at time required by the schedule, two robots should get close enough with each other to observe each other's behavior. The co-observation constraint is then modeled as a relative distance constraint between two robots at the given times (i.e., to require two agents see each other within a certain radius). The location of the co-observation location should make sure that the ellipsoidal reachability region between each co-observation, considered in later sections 4.4, has an empty intersection with all forbidden regions.

In this type of constraint, the robots are required to be in physical proximity of each other at some time instant j (e.g., to inspect each other, or to exchange data). We write the function for the constraint as:

$$D(\mathbf{q}) = \overrightarrow{q_{aj}q_{bj}}, \tag{4.17}$$

where a, b are the indices of the pair of agents required for a mutual inspection. The corresponding constraint set is simply

$$\zeta = \{z \mid z \leq d_{max}\}, \tag{4.18}$$

with a projection function:

$$\Pi_\zeta(z) = \begin{cases} d_{max}\frac{z}{\|z\|} & \text{if } \|z\| > d_{max}, \\ z & otherwise. \end{cases} \tag{4.19}$$

The locations q_{aj} and q_{bj} where the co-observation performed are computed as part of

the optimization.

4.4 Reachability constraints

In this section, we first provide the definition of an *ellipsoidal reachability region*. We then provide a differentiable map to transform such region in a canonical axis-aligned form where the operator Π_ζ and its differential can be obtained; these operators are then extended to the general case via the aforementioned transform. The overall goal is to define the functions $D(q)$, its differential, and the operator Π_ζ for *ellipsoidal reachability regions* with respect to the forbidden regions that can then be used in the ADMM formulation in Chapter 3.

4.4.1 Definition of reachability constraint

The reachability region is defined as the set of locations $q(t)$ that a robot can reach between two given fixed positions:

Definition 2. *The* reachability region *for two waypoints $q(t_1) = q_1$, $q(t_2) = q_2$ is defined as the sets of points q' in the workspace such that there exist a trajectory $q(t)$ where $q(t') = q'$, $t_1 \leq t' \leq t_2$ and $q(t)$ satisfies the velocity constraint $d(q(t), q(t+1)) \leq v_{max}$.*

This region can be analytically bounded via an ellipsoid:

Definition 3. *The* reachability ellipsoid *is defined as the region $\mathcal{E}(q_1, q_2, t_1, t_2) = \{\tilde{q} \in \mathbb{R}^n : d(q_1, \tilde{q}) + d(\tilde{q} + q_2) < 2a\}$, where $a = \frac{v_{maq}}{2}(t_2 - t_1)$.*

The region $\mathcal{E}(q_1, q_2)$ is an ellipsoid with foci at q_1, q_2, center $o_\mathcal{E} = \frac{1}{2}(q_1 + q_2)$, and the major radius equal to a. Let $c_\mathcal{E} = \frac{1}{2}\|q_1 - q_2\| = \|o_\mathcal{E} - q_1\|$ be the distance from the center to a foci.

The reachability ellipsoid is an over-approximation of the exact reachability region; the difference between the two is due to the discretization of the trajectory, and the fact that $\mathcal{E}$ does not consider the presence of obstacles.

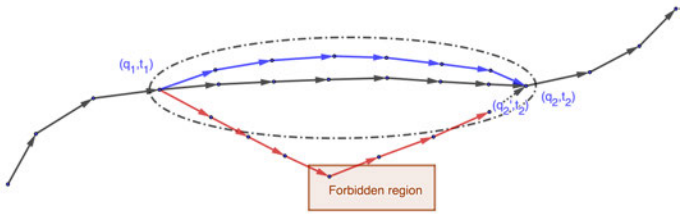


Figure 4·1: The ellipse showcases the reachability region. The black line is the planned trajectory of a robot, q_1 and q_2 are two locations this robot are expected at given time t_1 and t_2, blue and red line are possible trajectories if the robot is compromised in after reaching q_1. Robot that goes outside the reachability region to forbidden region can only reach q_2' instead of q_2 at time t_2, even if it drives directly toward q_2 at full speed v_{max} after the intrusion into forbidden region.

4.4.2 Transformation to canonical coordinates

To simplify the problem, a canonical rigid body transformation is used to transform the ellipse $\mathcal{E}$ from a global frame $\mathcal{F}$ to a canonical frame $\mathcal{F}_\mathcal{E}$. The latter is defined such that the center of the ellipsoid is located at the origin and the foci are aligned with the first axis of $\mathcal{F}_\mathcal{E}$. Since the transformation depends on the two foci, i.e., two waypoints of an agent, the challenge here is to construct the transformation in a differentiable way. For the convenience of derivation, we define the coordinate transformation from $\mathcal{F}$ to $\mathcal{F}_\mathcal{E}$ using a rotation $R_\mathcal{E}^\mathcal{F}$ and a translation $o_\mathcal{E}^\mathcal{F}$, which, to simplify the notation, from now on we simply refer to as R and o, respectively. The transformation of a point from the frame $\mathcal{F}_\mathcal{E}$ to the frame $\mathcal{F}$ and its inverse are given by the formulation:

$$q^\mathcal{F} = Rq^\mathcal{E} + o, \quad q^\mathcal{E} = R^\mathrm{T}(q^\mathcal{F} - o). \tag{4.20}$$

The reverse transformation is $q^\mathcal{E} = R^\mathrm{T}(q^\mathcal{F} - o)$.

We define $\nu_\mathcal{F}$ and $\nu_\mathcal{E}$ to represent the x-axis unitary vector of $\mathcal{F}_\mathcal{E}$ in the frames $\mathcal{F}$

and $\mathcal{F}_\mathcal{E}$, respectively. Formally:

$$\nu'_\mathcal{F} = q_2 - q_1, \qquad \nu_\mathcal{F} = \frac{\nu'_\mathcal{F}}{\|\nu_\mathcal{F}\|}, \qquad \nu_\mathcal{E} = [1,0,0]^T; \tag{4.21}$$

see Fig.4·1 for an illustration. Note that $\nu_\mathcal{E}$ is constant while, for the sake of clarity, $\nu_\mathcal{F}$ depends on q_1, q_2. We then define the rotation R using a Householder rotation, while from (4.20) we see that o represents the center of $\mathcal{E}(q_1, q_2)$ expressed in $\mathcal{F}$, i.e.:

$$R = H(\nu_\mathcal{F}(q_1,q_2), \nu_\mathcal{E}(q_1,q_2)), \qquad o = \frac{1}{2}(q_1 + q_2). \tag{4.22}$$

To simplify the notation, in the following we will consider H to be a function of q_1, q_2 directly, i.e. $H(q_1, q_2)$.

4.4.3 Reachability constraints via ellipsoids

In later sections, we define constraint formulation of the ellipsoid regions against different types of forbidden regions: a point, a plane, a segment, and a convex polygon. For each one, our goal is to define the function $D(q)$, the set ζ and the projection Π_ζ that can be incorporated in the ADMM optimization (3.3); we also include derivations for the differential $\partial_q D(q)$, which can be used to significantly speed up the optimization.

4.4.4 Point-ellipsoid constraint

As shown in Fig.4·2, we consider a forbidden region in the shape of a single point q_{avoid}. The goal is to design the trajectory $q(t)$ such that $q_{avoid} \notin \mathcal{E}(q_1, q_2)$. We first define a projection function $\pi_{p\mathcal{E}}(q_{avoid}; q_1, q_2, a) = q_p$, which returns a projected point q_p of q_{avoid} outside the ellipse, i.e., as the solution to

$$\begin{aligned} \underset{q_p}{\text{argmin}} \quad & \|q_{avoid} - q_p\|^2 \\ s.t. \quad & q_p \in \mathcal{E}^c. \end{aligned} \tag{4.23}$$

where $\mathcal{E}^c$ is the set complement of the region $\mathcal{E}$.

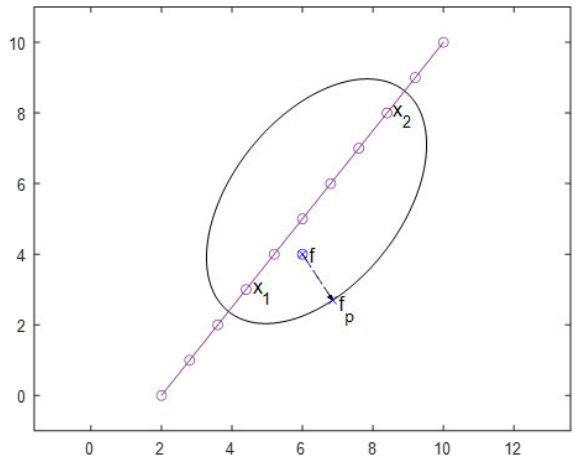


Figure 4·2: Point-ellipsoid constraint, a point f inside ellipsoid is projected to the areas outside the ellipsoid f_p.

Then, the constraint $q_{avoid} \notin \mathcal{E}(q(t_1), q(t_2))$ can be written as:

$$D(q) = \pi_{p\mathcal{E}}(q_{avoid}; q(t_1), q(t_2), r) - q_{avoid} = 0 \tag{4.24}$$

For cases where $q_{avoid} \notin \mathcal{E}(q(t_2), q(t_1), r)$, $\pi_{p\mathcal{E}}(q_{avoid}) = q_{avoid}$. And for cases where $q_{avoid} \in \mathcal{E}(q_1, q_2, r)$, $D(q)$ needs to be projected to the boundary of the ellipse, which is discussed below.

Projection to the standard ellipse

The ellipse $\mathcal{E}$ expressed in $\mathcal{F}_\mathcal{E}$ is given by $\mathcal{E}^\mathcal{E} = \{q^\mathcal{E} \in \mathbb{R}^m : d(q_1^\mathcal{E}, q^\mathcal{E}) + d(q^\mathcal{E}, q_2^\mathcal{E}) < 2a\}$, where the coordinates of the two foci $q_1^\mathcal{E}, q_2^\mathcal{E}$ in $\mathcal{F}_\mathcal{E}$ are

$$q_1^\mathcal{E} = \begin{bmatrix} c & 0 & 0 \end{bmatrix}^\mathrm{T}, \qquad q_2^\mathcal{E} = \begin{bmatrix} -c & 0 & 0 \end{bmatrix}^\mathrm{T}, \tag{4.25}$$

with $c = \frac{\|q_2 - q_1\|}{2}$.

The ellipsoid $\mathcal{E}$ in the canonical frame can be described as the zero level set of the quadratic function

$$E^{\mathcal{E}}(q^{\mathcal{E}}) = {q^{\mathcal{E}}}^{\mathrm{T}} Q q^{\mathcal{E}} - 1 \tag{4.26}$$

where

$$Q = \mathrm{diag}(a^{-2}, b^{-2}, b^{-2}), \tag{4.27}$$

and $b = \sqrt{a^2 - c^2}$. The ellipse parameters a,b represent the lengths of the major axes.

The point to project, q_{avoid}, can be likewise expressed in $\mathcal{F}_{\mathcal{E}}$ as ${q^{\mathcal{E}}}_{avoid} = H({q^{\mathcal{F}}}_{avoid} - o)$.

We now turn our attention to the problem of projecting ${q^{\mathcal{E}}}_{avoid}$ on the zero level set of $E^{\mathcal{E}}$ (i.e., the reachability ellipsoid in the canonical frame). The derivations below are loosely inspired by [Eberly, 2013].

Let $q_p^{\mathcal{E}}$ be the point on the surface of the ellipsoid, i.e., $E^{\mathcal{E}}(q_p^{\mathcal{E}}) = 0$, corresponding to the projection of the point $q_{avoid}^{\mathcal{E}}$. Using Lagrange multipliers applied to the constrained optimization problem (4.23) (after transforming it in the canonical frame), one can show that the vector from a point to its projection, $q_{avoid}^{\mathcal{E}} - q_p^{\mathcal{E}}$, must be collinear with the gradient of $\mathcal{E}^{\mathcal{E}}$, i.e.

$$q_p^{\mathcal{E}} - q_{avoid}^{\mathcal{E}} = s\partial_q E^{\mathcal{E}}(q_p^{\mathcal{E}})^{\mathrm{T}} = sQq_p^{\mathcal{E}} \tag{4.28}$$

for some scale $s \in \mathbb{R}$; thus $q_p^{\mathcal{E}}$ can be written as:

$$q_p^{\mathcal{E}} = (I + sQ)^{-1} q_{avoid}^{\mathcal{E}} = S q_{avoid}^{\mathcal{E}} \tag{4.29}$$

where $S = (I + sQ)^{-1}$. Using the fact that since $q_p^{\mathcal{E}}$ is a point on the ellipse, s can be solved as the root of the equation obtained by substituting (4.29) in $E^{\mathcal{E}}(q^{\mathcal{E}})$:

$$0 = F(s) = {q_p^{\mathcal{E}}}^{\mathrm{T}} Q q_p^{\mathcal{E}} - 1 = {q_{avoid}^{\mathcal{E}}}^{\mathrm{T}} Q'(s) q_{avoid}^{\mathcal{E}} - 1, \tag{4.30}$$

where

$$Q'(s) = S^{\mathrm{T}} Q S = \operatorname{diag}\left(\frac{a^2}{(s+a^2)^2}, \frac{b^2}{(s+b^2)^2}, \frac{b^2}{(s+b^2)^2}\right) \tag{4.31}$$

Detailed methods for computing s can be found in [Eberly, 2013].

Then the point-to-ellipse projection function can be represented as:

$$\begin{aligned} \pi_{p\mathcal{E}}(q) &= R^{-1}(q(t_1), q(t_2)) q_p^{\mathcal{E}} + o \\ &= R^{-1}(q(t_1), q(t_2)) S q_{avoid}^{\mathcal{E}} + o \\ &= R^{-1} S R (q_{avoid} - o) + o \end{aligned} \tag{4.32}$$

In our derivations, we consider only the 3-D case ($m = 3$); for the 2-D case, let $P = \begin{bmatrix} I & 0 \end{bmatrix} \in \mathbb{R}^{2\times 3}$: then $\pi_{p\mathcal{E}}^{\mathrm{2D}} = P \pi_{p\mathcal{E}}^{\mathrm{3D}}(P^{\mathrm{T}} q_{avoid}; P^{\mathrm{T}} q_1, P^{\mathrm{T}} q_2, a)$.

ADMM constraints

The corresponding constraint is written as

$$D_p(q) = \begin{cases} \pi_{p\mathcal{E}}(q) - q_{avoid} & q_{avoid} \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.33}$$

and feasible set and projection function as:

$$\zeta = \{q \in \mathbb{R}^{nm} : \|D_p(q)\| = 0\}, \quad \Pi(D_p(q)) = 0 \tag{4.34}$$

Differential of the constraint

Proposition 3. *The differential of the projection operator $\pi_{p\mathcal{E}}(q_{avoid}; q_1, q_2, a)$ with respect to the foci q_1, q_2 is given by the following (where we use q as a shorthand*

notation for $q^{\mathcal{E}}_{avoid}$)

$$\begin{aligned}\partial_{\left[\begin{smallmatrix}q_1\\q_2\end{smallmatrix}\right]}\pi_{p\mathcal{E}} = -2H[SH(q-o)]_\times U \\ + \big((q^{\mathrm{T}}\partial_s Q' q)^{-1} H^{-1} Q' q q^{\mathrm{T}} (4Q' H[q-o]_\times U \\ +2Q' H\partial_q o - \partial_b Q' q q \partial_q b) - sH^{-1}S^2\partial_b Q q \partial_q b\big) \\ - 2H^{-1}SH[q-o]_\times U + (H^{-1}SH - I)\partial_q o\end{aligned} \tag{4.35}$$

Proof. To make the notation more compact, we will use $\partial_q f$ instead of $\partial_{\left[\begin{smallmatrix}q_1\\q_2\end{smallmatrix}\right]} f$ for the remainder of the proof. The differential of (4.32) can be represented as:

$$\begin{aligned}\dot\pi_{p\mathcal{E}} =& \dot H^{-1} SH(q_{avoid}-o) + H^{-1}\dot S H(q_{avoid}-o) \\ &+ H^{-1}S\dot H(q_{avoid}-o) + (H^{-1}SH - I)\dot o\end{aligned} \tag{4.36}$$

where

$$\begin{aligned}\dot S =& -S^2(Q\dot s + s\dot Q) \\ =& -S^2(Q\partial_q s\dot q - \partial_b Q \partial_q b \dot q)\end{aligned} \tag{4.37}$$

where

$$\partial_b Q = 2\frac{s}{b^3}\,\mathrm{diag}\{0,1,1\} \tag{4.38}$$

To compute the derivative $\partial_q \pi$, we need the expression of $\partial_q b$, $\partial_q o$ and $\partial_q s$; the first two can be easily derived using the equations above:

$$\partial_q b = \frac{1}{4b}\left[q_1 - q_2, q_2 - q_1\right]^{\mathrm{T}} \tag{4.39}$$

$$\partial_q o = \left[I/2, I/2\right]^{\mathrm{T}} \tag{4.40}$$

In order to get $\partial_q s$, we use the fact that $F\big(s(q)\big) = 0$ for all q; hence $F\big(\tilde q(t)\big) \equiv 0$, and $\partial_q F = 0$. We then have:

$$0 = \dot F = 2q^{\mathrm{T}} Q' \dot q + q^{\mathrm{T}}\partial_s Q' q \dot s + q^{\mathrm{T}}\partial_b Q' q \dot b \tag{4.41}$$

where

$$\partial_s Q' = -\,\mathrm{diag}\left(\frac{2a^2}{(s+a^2)^3}, \frac{2b^2}{(s+b^2)^3}, \frac{2b^2}{(s+b^2)^3}\right). \tag{4.42}$$

By moving term $\dot{s}$ to the left-hand side we can obtain:

$$\begin{aligned}\dot{s} = (q^{\mathrm{T}}\partial_s Q' q)^{-1}(2q^{\mathrm{T}}Q'\dot{q} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \\ = (q^{\mathrm{T}}\partial_s Q' q)^{-1}(-4q^{\mathrm{T}}Q'H[U\dot{q}]_\times(q_{avoid} - o) \\ - 2q^{\mathrm{T}}Q'H\dot{o} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \\ = (q^{\mathrm{T}}\partial_s Q' q)^{-1}(-4q^{\mathrm{T}}Q'H[q_{avoid} - o]_\times U\dot{q} \\ - 2q^{\mathrm{T}}Q'H\dot{o} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \quad (4.43)\end{aligned}$$

The second term of equation (4.36) turns into:

$$\begin{aligned}H^{-1}\dot{S}H(q_{avoid} - o) = -H^{-1}Q' q\dot{s} - sH^{-1}S^2\partial_b Q q\dot{b} \\ = \big((q^{\mathrm{T}}\partial_s Q' q)^{-1}H^{-1}Q' q q^{\mathrm{T}}(4Q'H[q_{avoid} - o]_\times U \\ + 2Q'H\partial_q o - \partial_b Q' q q\partial_q b) - sH^{-1}S^2\partial_b Q q\partial_q b\big)\dot{q}\end{aligned} \quad (4.44)$$

Thus equation (4.36) could be written as:

$$\begin{aligned}\dot{\pi}_{p\mathcal{E}} = \big(- 2H[SH(q_{avoid} - o)]_\times U \\ + \big((q^{\mathrm{T}}\partial_s Q' q)^{-1}H^{-1}Q' q q^{\mathrm{T}}(4Q'H[q_{avoid} - o]_\times U \\ + 2Q'H\partial_q o - \partial_b Q' q q\partial_q b) - sH^{-1}S^2\partial_b Q q\partial_q b\big) \\ - 2H^{-1}SH[q_{avoid} - o]_\times U \\ + (H^{-1}SH - I)\partial_q o\big)\dot{q}, \quad (4.45)\end{aligned}$$

from which the claim follows. □

The differential of D_p is the same as the one for $\pi_{p\mathcal{E}}$.

4.4.5 Plane-ellipsoid constraint

For a intermediate step to get the relationship between a polygon shaped forbidden region, we first consider a forbidden region in the shape of a hyperplane $\mathcal{L}(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^n : \mathbf{n}^{\mathrm{T}}\tilde{q} = d\}$ (as shown in Figure 4·3). The reachability constraint now can then be defined as $\mathcal{L} \cap \mathcal{E}(q_1, q_2, a) = \emptyset$. Using the transformation introduced in (4.4.2), the

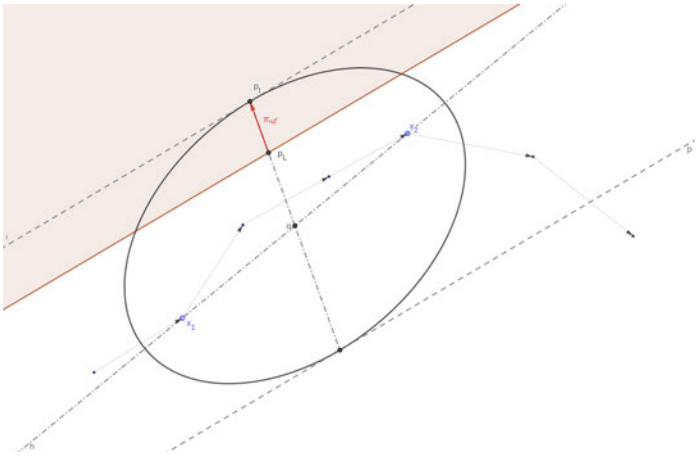


Figure 4·3: Plane-ellipse constraint. Projected a ellipsoid to one side of a plane. The projection is simplified to the point-ellipse constraint as projecting a point inside ellipse p_L to p_t on outside the ellipse.

ellipse can be transferred into a standard one with the new hyperplane in the form of $\mathcal{L}^{\mathcal{E}}(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_{\mathcal{E}}^{\mathrm{T}}\tilde{q} = d_{\mathcal{E}}\}$, where,

$$n_{\mathcal{E}} = H(q_1, q_2)n \tag{4.46}$$

$$d_{\mathcal{E}} = -\mathbf{n}^{\mathrm{T}}o + d \tag{4.47}$$

Projection via the tangent interpolation point

For each $\mathcal{L}_{\mathcal{E}}$, there exist two planes ${\mathcal{L}^{\mathcal{E}}}_1 = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_{\mathcal{E}}^{\mathrm{T}}\tilde{q} = d_{\mathcal{E}t}\}$ and ${\mathcal{L}^{\mathcal{E}}}_2 = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_{\mathcal{E}}^{\mathrm{T}}\tilde{q} = -d_{\mathcal{E}t}\}$ that are parallel to $\mathcal{L}$ and tangent to the ellipse, where

$$d_{\mathcal{E}t} = \sqrt{n_{\mathcal{E}}^{\mathrm{T}}Q^{-1}n_{\mathcal{E}}}. \tag{4.48}$$

The corresponding tangent points are given as

$$p_1^{\mathcal{E}} = \frac{d_{\mathcal{E}t}Q^{-1}n_{\mathcal{E}}}{n_{\mathcal{E}}^{\mathrm{T}}Q^{-1}n_{\mathcal{E}}} = \frac{Q^{-1}n_{\mathcal{E}}}{d_{\mathcal{E}t}}, \tag{4.49}$$

$$p_2^{\mathcal{E}} = -p_1^{\mathcal{E}}. \tag{4.50}$$

We use these two tangent points to define a novel measure of displacement between a plane and an ellipse.

Definition 4. *The* tangent interpolation point *is defined as*

$$p_{\mathcal{L}}^{\mathcal{E}} = \frac{d_{\mathcal{E}} Q^{-1} n_{\mathcal{E}}}{n_{\mathcal{E}}^{\mathrm{T}} Q^{-1} n_{\mathcal{E}}} \tag{4.51}$$

Note that $p_{\mathcal{L}}^{\mathcal{E}} \in \mathcal{L}$ and when $d_{\mathcal{E}} = d_{\mathcal{E}t}$ or $d_{\mathcal{E}} = -d_{\mathcal{E}t}$, $p_{\mathcal{L}}^{\mathcal{E}} = p_1^{\mathcal{E}}$ or $p_{\mathcal{L}}^{\mathcal{E}} = p_2^{\mathcal{E}}$, respectively. It is clear that when $d_{\mathcal{E}} \in [-d_{\mathcal{E}t}, d_{\mathcal{E}t}]$, the plane $\mathcal{L}$ and the ellipsoid $\mathcal{E}$ have at least one intersection, thus violating our desired reachability constraint. Using this fact, we define the following projection operator:

$$\pi_{\mathbf{n}\mathcal{E}}^{\mathcal{E}}(q) = \begin{cases} p_{t1}^{\mathcal{E}} - p_{\mathcal{L}}^{\mathcal{E}} & d_{\mathcal{E}} \in [0, d_{\mathcal{E}t}], \\ p_{t2}^{\mathcal{E}} - p_{\mathcal{L}}^{\mathcal{E}} & d_{\mathcal{E}} \in [-d_{\mathcal{E}t}, 0), \\ 0 & \text{otherwise.} \end{cases} \tag{4.52}$$

ADMM constraints

Transforming the projection $\pi^{\mathcal{E}}$ back to $\mathcal{F}$ we have:

$$D_{\mathbf{n}}(q) = H^{-1}(q(t_1), q(t_2)) \pi_{\mathbf{n}\mathcal{E}}^{\mathcal{E}}(q) + o, \tag{4.53}$$

with the corresponding constraint written as

$$\zeta_{\mathbf{n}} = \{q \in \mathbb{R}^{nm} : \|D_{\mathbf{n}}(q)\| = 0\} \tag{4.54}$$

$$\Pi_{\mathbf{n}}(D_{\mathbf{n}}(q)) = 0 \tag{4.55}$$

Gradients for the projection function

Proposition 4. *The differential of the projection function* $\Pi^{\mathcal{E}}_{\mathbf{n}\mathcal{E}}(q)$ *with respect to the foci* q_1 *and* q_2 *is given by:*

$$\partial_q \pi^{\mathcal{E}}_{\mathbf{n}\mathcal{E}}(q) = \begin{cases} \partial_q p^{\mathcal{E}}_{t1} - \partial_q p^{\mathcal{E}}_{\mathcal{L}} & d_{\mathcal{E}} \in [0, d_{\mathcal{E}t}], \\ \partial_q p^{\mathcal{E}}_{t2} - \partial_q p^{\mathcal{E}}_{\mathcal{L}} & d \in [-d_{\mathcal{E}t}, 0), \\ 0 & otherwise. \end{cases} \tag{4.56}$$

where

$$\partial_q p_{\mathcal{L}} = (-\frac{d_{\mathcal{E}t} n^{\mathrm{T}} \partial_q o - 2 d_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d^3_{\mathcal{E}t}}) Q^{-1} n_{\mathcal{E}} + \frac{d_{\mathcal{E}} \partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2 d_{\mathcal{E}} Q^{-1} H[n]_\times U}{d^2_{\mathcal{E}t}}, \tag{4.57}$$

$$\partial_q p_1 = -\frac{Q^{-1} n_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d^2_{\mathcal{E}t}} + \frac{\partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2 Q^{-1} H[n]_\times U}{d_{\mathcal{E}t}}. \tag{4.58}$$

Proof. We first need to derive $\dot{d}_{\mathcal{E}}$ and $\dot{d}_{\mathcal{E}t}$

$$\dot{d}_{\mathcal{E}} = -n^{\mathrm{T}} \partial_q o \dot{q} \tag{4.59}$$

$$\begin{aligned} \dot{d}_{\mathcal{E}t} =& (\dot{n}^{\mathrm{T}}_{\mathcal{E}} Q^{-1} n_{\mathcal{E}} + n^{\mathrm{T}}_{\mathcal{E}} \dot{Q}^{-1} n_{\mathcal{E}} + n^{\mathrm{T}}_{\mathcal{E}} Q^{-1} \dot{n}_{\mathcal{E}}) / \sqrt{n^{\mathrm{T}}_{\mathcal{E}} Q^{-1} n_{\mathcal{E}}} \\ =& (\sqrt{n^{\mathrm{T}}_{\mathcal{E}} Q^{-1} n_{\mathcal{E}}})^{-1} \left(-2 n^{\mathrm{T}} H [Q^{-1} n_{\mathcal{E}}]_\times U \right. \\ & \left. + n^{\mathrm{T}}_{\mathcal{E}} \partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2 n_{\mathcal{E}} Q^{-1} H[n]_\times U \right) \dot{q} \end{aligned} \tag{4.60}$$

Next, we need to derive $\dot{p}_{t1}$, $\dot{p}_{t2}$ and $\dot{p}_{\mathcal{L}}$. Since $p_{\mathcal{L}}$ could be written as

$$p_{\mathcal{L}} = \frac{d_{\mathcal{E}} Q^{-1} n_{\mathcal{E}}}{d^2_{\mathcal{E}t}}, \tag{4.61}$$

we have

$$\dot{p}_{\mathcal{L}} = \left((-\frac{d_{\mathcal{E}t} n^{\mathrm{T}} \partial_q o - 2 d_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d^3_{\mathcal{E}t}}) Q^{-1} n_{\mathcal{E}} + \frac{d_{\mathcal{E}} \partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2 d_{\mathcal{E}} Q^{-1} H[n]_\times U}{d^2_{\mathcal{E}t}} \right) \dot{q} \tag{4.62}$$

$$\dot{p}_1 = \left(-\frac{Q^{-1} n_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^2} + \frac{\partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2Q^{-1} H[n]_\times U}{d_{\mathcal{E}t}} \right) \dot{q} \quad (4.63)$$

subtracting $\dot{q}$ from (4.62) and (4.63), we can derive the result shown in (4.51) □

Based on the previous proposition, the differential of (4.53) can be written as:

$$\partial_q D_{\mathbf{n}\mathcal{E}} = -2H[\Pi^{\mathcal{E}}{}_{\mathbf{n}\mathcal{E}}]_\times M + H^{-1} \partial_q \Pi^{\mathcal{E}}{}_{\mathbf{n}\mathcal{E}} \quad (4.64)$$

4.4.6 Line-segment-ellipse constraint

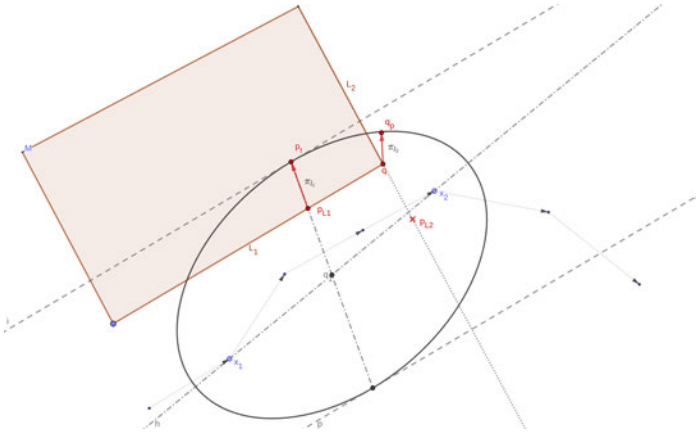


Figure 4·4: Polygen-ellipse constraint. This constraint is simplified to either a plane-ellipse constraint or a point-ellipse constraint.

To avoid (or stay in) a region defined by several hyperplanes, the relative position between the ellipse and segment of the hyperplane needs to be studied instead of the whole plane. Assume the segment has endpoints p_1 and p_2, the segment can be defined as:

$$\begin{bmatrix} (p_1 - p_2)^{\mathrm{T}} \\ (p_2 - p_1)^{\mathrm{T}} \end{bmatrix} p \leq \begin{bmatrix} p_2^{\mathrm{T}} \\ -p_1^{\mathrm{T}} \end{bmatrix} (p_1 - p_2) \quad (4.65)$$

And using the $p_{\mathcal{L}}$ defined in (4.51), we define

$$D_{seg}(q) = \begin{cases} D_1(p_1) & (p_1 - p_2)^{\mathrm{T}}(p_{\mathcal{L}} - p_2) < 0 \\ D_1(p_2) & (p_2 - p_1)^{\mathrm{T}}(p_{\mathcal{L}} - p_1) < 0 \\ D_2(p_{\mathcal{L}}) & otherwise \end{cases} \tag{4.66}$$

where $D_{p\mathcal{E}}$ and $D_{\mathbf{n}\mathcal{E}}$ are corresponding constraint function introduced in section 4.4.4 and 4.4.5 with the constraint set and projection written as:

$$\zeta_{seg} = \{q \in \mathbb{R}^{nm} : \|D_{seg}(q)\| = 0\} \tag{4.67}$$

$$\Pi_{seg}(D_{seg}(q)) = 0 \tag{4.68}$$

4.4.7 Convex-polygon-ellipse constraint

To keep an ellipse away from a convex polygon, first, we need to keep all segments of the hyperplanes outside the ellipse. And a convex obstacle constraint for the polygon (as introduced in section 3.3.4) is also added to prevent cases where the ellipse is a subset of the region. Similar to (4.51), we define

$$D_{cp}(q) = \begin{bmatrix} D_{seg1} \\ D_{seg2} \\ \vdots \end{bmatrix} \tag{4.69}$$

with the constraint set and projection written as:

$$\zeta_{cp} = \{q \in \mathbb{R}^{nm} : \|D_{cp}(q)\| = 0\} \tag{4.70}$$

$$\Pi_{cp}(D_{cp}(q)) = 0 \tag{4.71}$$

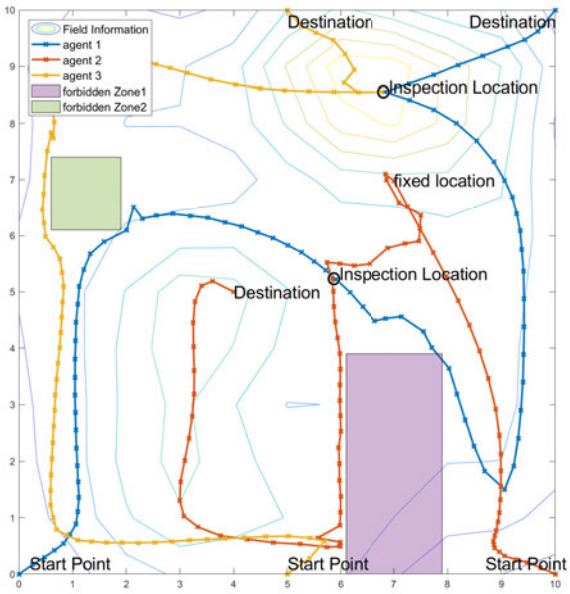


Figure 4·5: Additional constraints added on top of 3·2 to testify the constraints, including to introspection (co-observaiton) constraint, location with fixed deadline constraint.

4.5 Result and simulation

In this section, we first test the new constraints using the same setup as in section 3.4, where we added a waypoint constraint for the location at $[7, 7]$ (at any point in the mission), and two co-observation constraint requiring robot 1 and robot 2 to meet at time 40, robot 1 and robot 3 to meet at time 70. The result can be seen in Figure 4·5 with resulting entries of the information matrix having a max of 10.0 and a min of 1.3.

4.5.1 Co-observation schedule and reachability constraints

In this section, we utilize the APMAPF solver introduced in [Wardega et al., 2019] to generate a MAPF plan with co-observation schedule on a grid-world application and transform the result to a continuous configuration space, under the map exploration

tasks. The environment is a $8m \times 8m$ region with an obstacle *Zone 1* and two forbidden regions *Zone 2, 3* which is shown in Fig.4·6. The robots have a maximum velocity of $0.5m/dt$ and a time limit of 20. All robots have the task of collecting sensor information on the underlying vector field, with agent 3 having an additional task to visit the *checkpoint* at time 8. Assuming that the sensors on the robots return data with higher accuracy for locations closer to the agent, the robots should ideally perform a boustrophedon pattern (like the unconstraint result in Figure 3·1). We first set up an co-observation schedules considering two forbidden regions using the APMAPF algorithm, and add reachability constraints ensure a empty intersection between all robots' reachability regions between co-observations and forbidden regions.

The co-observation schedule generated in a grid world is shown in Fig.4·6. No task has been assigned to the agent besides the start and goal location. Agent 1 and 2 need to meet at time 8 and 14, whereas agent 2 and 3 need to meet at time 18. This result is also used as the initial trajectory input for the ADMM solver. Based on the schedule, reachability for agent 1 in-between time 0, 8, 14 and 20, agent 2 in-between time 0, 8, 18 and 20, and agent 3 in-between time 0, 8, 18 and 20 are constrained. Notice that additional checkpoint (i.e. stationary security camera) need to be incorporated in order to generated an attack-proof solution. Additional solutions to avoid these checkpoints will be discussed in Chapter 5.

The result of the simulation is shown in Fig.4·7. The reachability regions are shown as black ellipsoids, and we can show that all of them have empty intersections with Zone 2 and 3. Notice that no explicit constraints have been enabled between reachability regions and obstacles, because we assume that robots have the basic obstacle avoidance capability and can not go through any hard obstacles. Therefore, the intersections between obstacles and ellipsoids, i.e. the case here between agent 3 and zone 1, are tolerable. All constraints have been satisfied and, to the best of

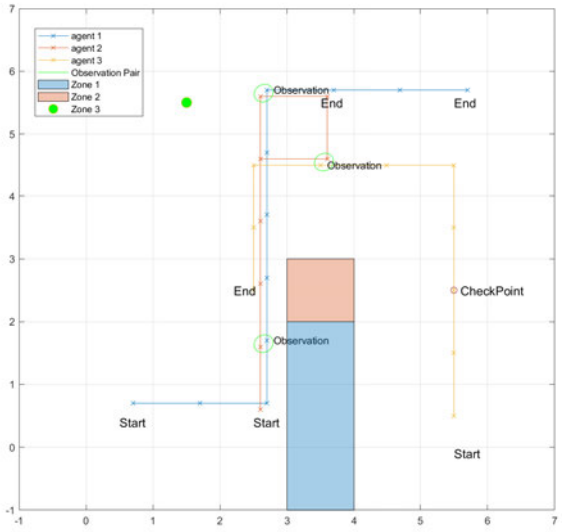


Figure 4·6: Observation schedule and initial trajectory generated in a 8×8 grid world. Zone 1 is obstacle, Zone 2 and Zone 3 are safe zones.

their capability, agents have spread out across the map to perform the best possible exploration task.

4.5.2 Experiments

The simulation result in Figure 3·2 has been selected for experimental validation using a team of three real robots. The goal of this experiment is to see if the plans can be followed despite the mismatch between the approximated behavior used in our algorithm and the actual robots. Our platform consists of three modified ground robots, iRobot Create2, that are controlled with a Raspberry Pi, and tracked using an OptiTrack Motion Capture System. For visual reference, we use short-throw projectors to visualize the environment and the paths. Each robot is controlled by an embedded PD controller which is fed the sequence of waypoints given by the planner. The full setup is shown in Figure 4·8. We verified, by inspection, that the robots successfully followed the waypoints designed by our algorithm while meeting the observation

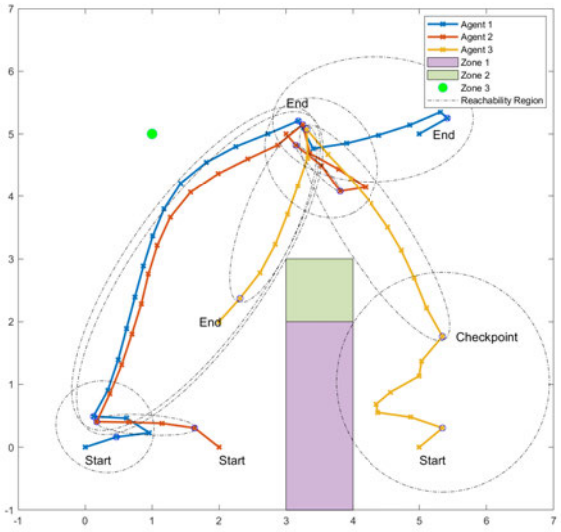


Figure 4·7: Simulation result for map exploration task. Reachability regions are shown as black ellipses in the result. Zone 1 is obstacle, Zone 2 and Zone 3 are safe zones

constraint.

4.6 Summary

In this chapter, we have incorporated co-observation schedule and the reachability constraints in our ADMM based algorithm. This constraint can be used to enhance the security of the system while still fulfilling the task. Our solution shows that planning with reachability and co-observation is a promising approach to enhance the security of multi-agent systems. However, two main challenges have been identified in [Wardega et al., 2023]. Firstly, finding a co-observation and reachability secured plan may not always be feasible when robots are separated from others by obstacles or forbidden regions, or are located far away from each other. In such case, it is impossible for other robots to get close enough to establish co-observation schedules or find a reachability-secured path. Agent 3 in the Figure 4·7 is a good example,

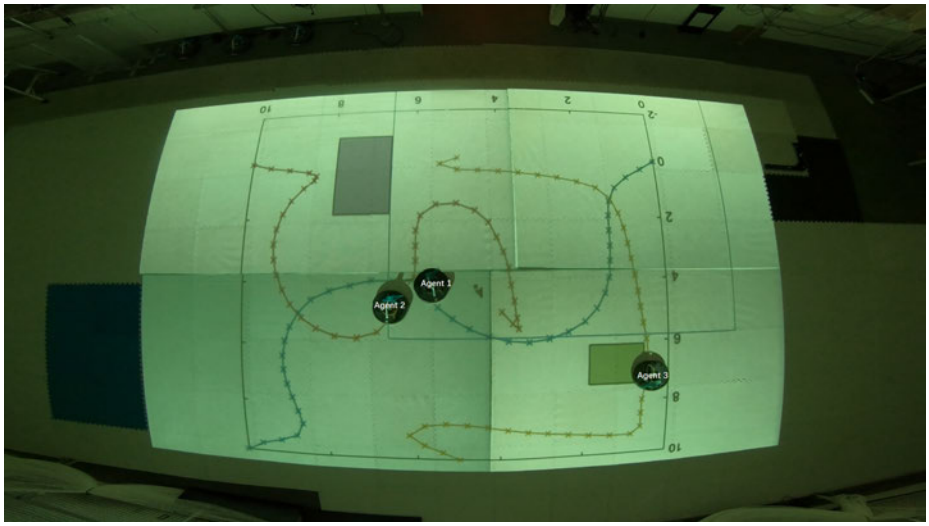


Figure 4·8: Experiment of a team of three robots where agent 1 and agent 2 meet at the pre specified time and locationh

as it required additional security checkpoint to create secured reachability areas. Secondly, security requirements may come at the cost of overall system performance, as illustrated by the comparison of Figure 3·2 and 4·7 where, after the introduction of security constraints, the top left corner is never explored by any robots. This is particularly concerning given that system performance is a key factor in the decision to use multi-agent systems. These challenges are addressed in Chapter 5, in order to ensure the effectiveness of planning with reachability and co-observation in securing multi-agent systems.

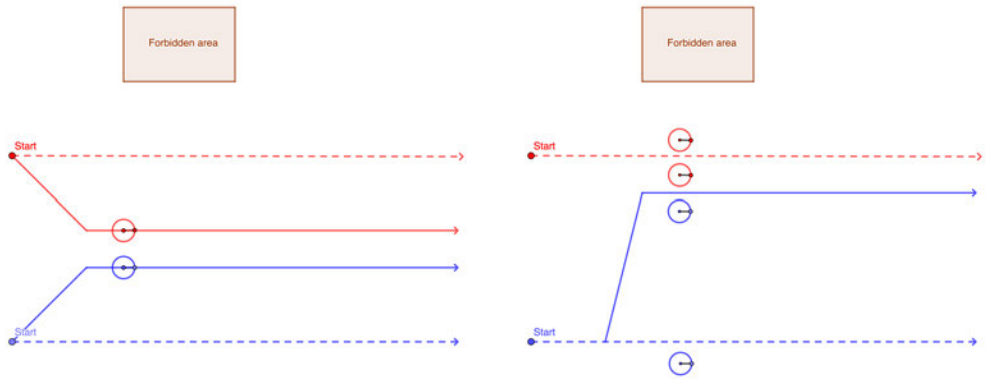
Chapter 5

Cross-trajectory Co-observation Planning using Sub-teams of Robots

In this chapter, we propose a solution to address the feasibility challenge and performance trade-off of the ADMM problem discussed in Section 4.6 by incorporating additional robots.

5.1 Co-observation planning using sub-teams

We propose to form *sub-teams* on each route, and setup additional co-observations both within the sub-team and across different sub-teams. Notice that co-observations across sub-teams (named *cross-trajectory co-observation*) are always preferred when feasible, since they provide better security in potential situations where the entire sub-team is compromised. In this chapter, we consider only the high-level path planning, while the assignment of the duties in the team will be performed dynamically as introduced later in Chapter 6), making it more difficult for attackers to successfully plan and execute attacks. Cross-trajectory co-observations allows sub-teams to preserve the optimal unsecured trajectories (as shown in Figure 5·1), as it does not require the entire *sub-team* to maintain a close distance with other teams when co-observation occurs.



(a) Co-observation **(b)** Cross-trajectory co-observation

Figure 5·1: 5·1a Limited by the co-observation requirement, both red and blue robot follow the co-observation secured routes (solid lines) and abandon the optimal ones (dashed line). 5·1b Through cross-trajectory co-observations, the blue team sends one robot to follow the red team (solid blue line) and performs co-observation while having the rest of the robots following the optimal trajectory.

5.2 Preliminaries

5.2.1 Rapidly-exploring Random Trees

We use RRT* as a component of the solution [Karaman and Frazzoli, 2010], which is an optimized variant of the rapidly-exploring Random Trees (RRT). As a optimal path planning algorithm, RRT* returns the shortest paths between an initial location and multiple potential goals. We assume that the generated paths can be travelled in both directions (this is used later in our analysis). The basic RRT* algorithm is shown in Algorithm 2. For a given tree $G = (V, E)$, key functions called by the algorithm are

`Sample`(i) This function returns independent identically distributed samples from the free configuration space.

`Nearest`(x) This function returns the vertex $v \in V$ that is closest in Euclidean distance to the input point x.

`Steer`(x, y) This function returns a point z that minimize $\|z-y\|$ subject to $\|z-x\| \leq \eta$ for a prespecified $\eta > 0$.

`ObstacleFree`(x, y) This function returns whether the line segment between x and y is free of collision.

`Cost`(v) This function assigns a non-negative cost (total travel distance in our application) to the unique path from the initial position to v .

`Parent`(v) This is a function that maps the vertex v to $v' \in V$ such that $(v', v) \in E$

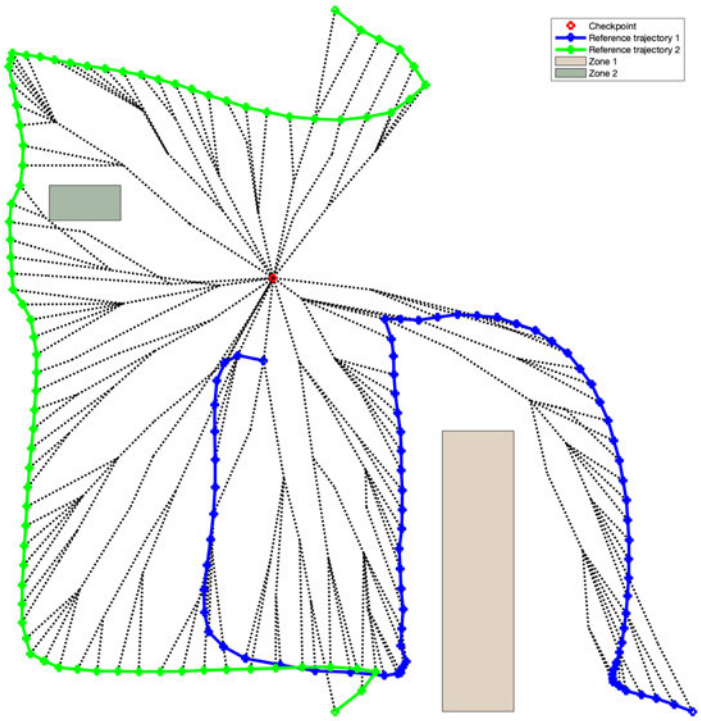


Figure 5·2: RRT* used to find trajectory from a point to different trajectories

Algorithm 2 RRT*$(x_{init}, \{x_{goal}\})$

$V_{goal} \leftarrow \{x_{goal}\}$
$V \leftarrow x_{init} \cup V_{goal}$, $E \leftarrow 0$
$i \leftarrow 0$
while $i < N$ **do**
    $G \leftarrow (V, E)$
    $x_{rand} \leftarrow \texttt{Sample}(i)$; $i = i + 1$;
    $x_{nearest} \leftarrow \texttt{Nearest}(G = (V, E), x_{rand})$;
    $x_{new} \leftarrow \texttt{Steer}(x_{nearest}, x_{rand})$;
    if $\texttt{ObstacleFree}(x_{nearest}, x_{new})$ **then**
        $V \leftarrow V \cup \{x_{new}\}$;
        $x_{min} \leftarrow x_{nearest}$;
        for all $x_{near} \in X_{near}$ **do**
            if $\texttt{ObstacleFree}(x_{near}, x_{new})$ **then**
                $c' \leftarrow \texttt{Cost}(x_{near}) + c(\texttt{Line}(x_{near}, x_{new}))$;
                if $c' < \texttt{Cost}(x_{new})$ **then**
                    $x_{min} \leftarrow x_{near}$;
                end if
            end if
        end for
        for all $x_{near} \in X_{near} \backslash x_{min}$ **do**
            if $\texttt{ObstacleFree}(x_{new}, x_{near})$ and $\texttt{Cost}(x_{near}) > \texttt{Cost}(x_{new} + c(\texttt{Line}(x_{new}, x_{near})))$ **then**
                $x_{parent} \leftarrow \texttt{Parent}(x_{near})$;
                $E' \leftarrow E' \backslash \{(x_{parent}, x_{near})\}$;
                $E' \leftarrow E' \cup \{(x_{new}, x_{near})\}$
            end if
        end for
    end if
end while
return V_{goal};

5.3 Problem overview

Assume we have a total of N robots in the system. We define a partition $\mathcal{I} = \cup_p \mathcal{I}_p$ of the robots such that robots in *sub-team* $\mathcal{I}_p$ have the same nominal trajectory. We use the planner introduced in Chapter 3 to generate multi-robot trajectories with $N_p < N$ routes $\{q_p\}_{p=1}^{N_p}$ without reachability and co-observation constraints, where the state $q_p(t), p \in \{0, \ldots, N_p\}$ represents the reference position of the i-th sub-team at time $t \in \{0, \ldots, t_e\}$. Within each sub-team, at least one robot is required to follow the reference trajectory to fulfill the required tasks. Meanwhile, the additional robots can switch from one reference trajectory to another in order to perform co-observation with different sub-teams. We refer to the new type of co-observations as *cross-trajectory co-observation*s. The goal is to find the cross trajectory co-observation plan to secure an unsecured MRS trajectory as with a minimal number of additional robots.

To solve this problem, we model the planned trajectory $\{q_p\}_{p=1}^{N_p}$ as a directed *checkpoint graph* $G_q = (V_q, E_q)$. Vertices V_q are a key locations in $\{q_p\}$ that are used to perform co-observations. Part of these vertices are *checkpoints* which are selected to guarantee that, if robots deviate and reach forbidden regions, they will miss the scheduled co-observations. Other vertices are selected to connect checkpoints on different trajectories. The set $E_q = E_t \cup E_c$ consists of directed edges of two types, trajectory edges E_t and cross-trajectory edges E_c, representing paths which robots can take. The trajectory edges E_t represent the reference trajectories, where each edge connects two waypoints on the trajectory of the same sub-team. The cross-trajectory edges E_c connect vertices on different reference trajectories, with at least one of the vertices being a checkpoint. These edges represent the potential paths that robots can take to travel and perform co-observations between trajectories of different sub-teams. Similar to the work done by [Yu and LaValle, 2013], we show that additional robots in the sub-team can be formulated as flows in the checkpoint graph, transforming the

co-observation planning problem into a network multi-flow problem that can be solved using general linear programming techniques as specialized solver.

In this chapter, we present a solution to the cross-trajectory co-observation planning problem. This approach consists of two components: constructing the checkpoint graph based on unsecured multi-robot trajectories, which includes the location of security checkpoints and cross-trajectory paths, and the formulation of the network flow problem to solve for the co-observation plan.

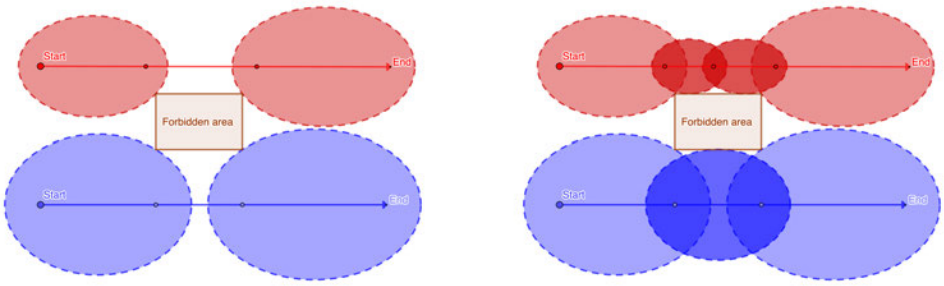
5.4 Checkpoint graph construction

In this section, we describe in detail how we define and search for security checkpoints and use the checkpoints to find cross-trajectory edges and construct a directed security graph $G_q = (V_q, E_q)$.

5.4.1 Checkpoints

Let $q \in \mathbb{R}^{nmT}$ be the MRS trajectory found in Chapter 3 for m sub-teams with time horizon $T = t_e$, with $q_p(t)$ be the reference trajectory waypoint for sub-team p at time t. A checkpoint $v_i = (q_i, t_i)$ is a pair composed of a location $q_i = q_p(t_i)$ and a time t_i representing a co-observation between a sub-team $\mathcal{I}_p$'s member and robots either from the same sub-team or different ones. For convenience, let $\mathcal{I}_{v_i}$ represent the corresponding sub-team v_i belongs to.

Similar to the requirements of co-observations introduced in Chapter 4, checkpoints $V_p = \{v_{p_1}, \ldots, v_{p_e}\} \subset V_q$ for a reference trajectory $\{q_p(t)\}$ of sub-team p need to guarantee that the reachability region between consecutive checkpoints $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}})$ does not intersect with any of the forbidden areas. For convenience, we denote the union of all forbidden areas as F; then the requirement can be written as $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}}) \cap F = \emptyset$, for every i.



(a) **(b)**

Figure 5·3: Checkpoint generation example

We provide a heuristic approach to locate the checkpoints on given trajectories; an optimal solution would likely be NP-hard, while the approach below works well enough for our purpose. We begin by adding the start and end locations $q_p(t_0)$, $q_p(t_e)$ to V_q, where $t_0 = 0$, and t_e is the time horizon. Then, we search for the waypoint $q_p(t_1)$ with the largest $t_1 \in \{t_0, \ldots, t_2\}$ such that the reachability ellipsoid between $q_p(t_0)$ and $q_p(t_1)$ has no overlap with any of the forbidden areas, i.e. $\mathcal{E}(q_p(t_0), q_p(t_1), t_0, t_1) \cap F = \emptyset$. Simultaneously, we search backward to find $q_p(t_3)$ with the smallest $t_3 \in \{t_1, \ldots, t_2\}$ that meets the reachability requirement $\mathcal{E}(q_p(t_2), q_p(t_3), t_2, t_3) \cap F = \emptyset$. $q_p(t_1)$ and $q_p(t_3)$ are then added to V_q. Afterwards, we set $t_0 = t_1$, $t_2 = t_3$ and repeat the same process. The search is stopped when $t_0 > t_2$, or when $\mathcal{E}(q_p(t_0), q_p(t_2), t_0, t_2) \cap F = \emptyset$. A toy example is shown in Figure 5·3.

5.4.2 Cross-trajectory edges

After locating checkpoints $\cup_p V_p$ for all sub-teams, we can search for cross-trajectory edges to connected all checkpoints to trajectories. Cross-trajectory edges represents feasible paths between two references trajectories that allow robots to reach a different reference trajectories and perform co-observations with the corresponding sub-teams.

Thus, at least one end of the edges must be a security checkpoint $v_p \in \cup_p V_p$. Additionally, assuming that only one robot is sent at a time between trajectories, the cross-trajectory edges must satisfy the reachability constraints from Chapter 4 to ensure that no deviation to forbidden areas can be performed while switching trajectories.

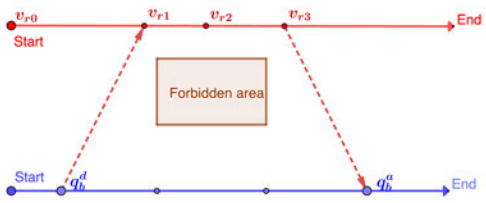


Figure 5·4: For checkpoints of the red trajectory, latest departure node q_b^d found for v_{r1} and earliest arrival node q_b^a found for v_{r3}.

To find cross-trajectory edges, we base our approach on RRT*. First, we search for feasible paths between each security checkpoint in each reference trajectory and all other trajectories, as shown in Figure 5·2. Since paths returned by RRT* are optimal and bidirectional, we can analyze the travel time and reachability for each candidate cross-trajectories, and add these edges to the checkpoint graph if they are feasible.

More vigorously, After a fixed iterations of RRT*, we find all possible and close to optimal paths between one security checkpoint $v_p = (q_p, t_p) \in V_p$ for sub-team $\mathcal{I}_p$, and $V_r = \{(q_{r_i}, t_{r_i})\}$ where $q_{r_i} = q_r(t_{r_i})$ are the entire waypoints on a reference trajectories for a different sub-team $\mathcal{I}_r$ for a fixed number of RRT* iterations. We can calculate the minimal travel time $t = \texttt{Cost}(q)/v_{max}$ for a robot to traverse each path. We call all nodes $v_{r_i} \in V_r$ *arrival nodes* if there exist a path between v_{r_i} and v_p with $t_p + t < t_{r_i}$, and the reachability ellipsoid $\mathcal{E}(q_r(t_{r_i}), q_p, t_{r_i}, t_p) \cap F = \emptyset$. Similarly, *departure nodes* are defined for v_{r_i} with $t_p > t_{r_i} + t$ and $\mathcal{E}(q_p, q_r(t_{r_i}), t_p, t_{r_i}) \cap F = \emptyset$. Given a point

$v_p \in V_p$, we define the *latest departure node* $q_r(t_r^d)$, and the *earliest arrival node* $q_r(t_r^a)$ as follows:

Latest departure node $q_r(t_r^d)$: is the waypoint with the latest time t_r^d such that a robot from sub-team $\mathcal{I}_r$ can meet with robots in sub-team $\mathcal{I}_p$ at v_p.

Earliest arrival node $q_r(t_r^a)$: is waypoint with the earliest time t_r^a such that a robot from sub-team $\mathcal{I}_p$ deviating from v_p can meet with sub-team $\mathcal{I}_r$.

For $v_p \in V_p$, latest departure node $v_d = (q_r(t_r^d), t_r^d)$ and earliest arrival node $v_a = (q_r(t_r^a), t_r^a)$, if feasible, are added to V_q. Cross-trajectory edges $(v_p \to v_a)$ and $(v_d \to v_p)$ are added to E_q. Examples are shown in Figure 5·4. Full algorithm is shown in Algorithm 3.

5.4.3 In-trajectory edges

All checkpoints V_q found in Section 5.4.1 and 5.4.2, can be divided by the corresponding sub-team and sorted in ascending order of time into N_p subsets $V_p = \{v_0^p, v_1^p, \ldots, v_T^p\}$ where $p \in \{1, \ldots, N_p\}$. Consecutive edges in the same subset are added to E_q as in-trajectory edges $(v_i^p \to v_{i+1}^p)$. Example are shown in Figure 5·5.

5.5 Co-observation planning problem

In this section, we describe in detail how to formulate the cross-trajectory planning problem as a network flow problem, and solve it using mixed-integer linear programs (MILP). We assume that there are always *reference robot*s in each sub-team following the reference trajectory. We want to plan the routes for additional *cross-trajectory robots* to perform co-observation with the reference robots.

To formulate the planning problem as a network multi flow problem, we augment the checkpoint graph G_q to a flow graph $G = (V, E)$. The vertices of the new graph are

Algorithm 3 Cross-trajectory edges generation

procedure CROSSTRAJECTORYEDGES(V)
  $V_q \leftarrow V$; $E_q \leftarrow 0$;
  for all $v \in V$ **do**
    for all $\mathbf{q}_r$ that $\mathcal{I}_r \neq \mathcal{I}_v$ **do**
      $V_q, E_q \leftarrow$ ADDEDGES$(q_v, \mathbf{q}_r, V_q, E_q)$;
    end for
  end for
  return V_q, E_q
end procedure
procedure ADDEDGES($q_v, \mathbf{q}_r, V_q, E_q$)
  $t_r^d \leftarrow -\infty$; $t_r^a \leftarrow \infty$;
  $\mathcal{V}_{goal} = RRT^*(q_v, \mathbf{q}_r)$;
  for all $q \in \mathcal{V}_{goal}$ that $\texttt{Parent}(q) \neq \emptyset$ **do**
    if $\texttt{Cost}(q)/v_{max} < t_q - t_v$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ **then**
      if $t_q < t_r^a$ **then**
        $t_r^a \leftarrow t_q$;
      end if
    else if $\texttt{Cost}(q)/v_{max} < t_v - t_q$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ **then**
      if $t_q > t_r^d$ **then**
        $t_r^d \leftarrow t_q$;
      end if
    end if
  end for
  if $t_r^d \neq -\infty$ and $t_r^a \neq \infty$ **then**
    $v_a \leftarrow (q(t_r^a), t_r^a)$; $v_d \leftarrow (q(t_r^d), t_r^d)$;
    $V_q \leftarrow V_q \cup v_a$; $V_q \leftarrow V_q \cup v_d$;
    $E_q \leftarrow E_q \cup (v_p \rightarrow v_a)$; $E_q \leftarrow E_q \cup (v_d \rightarrow v_p)$
  end if
  return V_q, E_q
end procedure

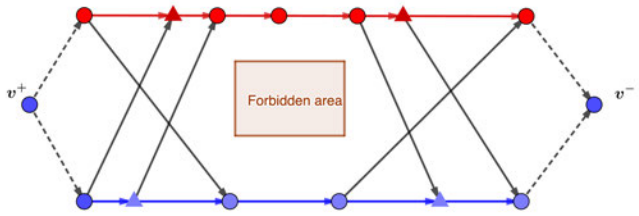


Figure 5·5: Example of a two trajectory security graph. Round vertices are checkpoint generated through the heuristic search and triangle vertices are added with the cross-trajectory edges. Additional virtual source v^+ and sink v^- is used later in planning problem.

defined as $V = V_q \cup \{v^+, v^-\}$, a *virtual source* v^+ and a *virtual sink* node v^- are added to V. The edges of the new graph are defined as $E = E_q \cup \{(v_p^e, v_-)\}_p \cup \{(v_+, v_p^0)\}_p$, where directed edges are added to E from v^+ to all the start vertices, and from all end vertices to v^-, with v_p^0 and v_p^e representing the start and end vertices of sub-team $\mathcal{I}_p$.

Assume that all the robots starting from the virtual source v^+ and end in the virtual sink v^-. Each robot may move from vertex v_i to v_j if $(v_i, v_j) \in E_q$. The path of a robot k can be represented as a flow vector $\mathbf{f}^k = \{f_{ij}^k\}$, where $f_{ij}^k \in \{1, 0\}$ is an indicator variable representing whether robot k's path contains the edge (v_i, v_j). As required by the security guarantee, reference robots must be seen by at least one surveillance robot at every checkpoint. Thus, the planning problem can then be formulated as a path cover problem on G_q, i.e., as finding a set of paths $F = [\mathbf{f}_1, \ldots, \mathbf{f}_\mathcal{K}]$ for cross-trajectory robots such that every checkpoint in V_p is included in at least one path in F. Additionally, to encourage the robots' exchange between different sub-teams, cross-trajectory co-observations are preferred compared to co-observation

within the same team. This is achieved with the weights for edges $(v_i, v_j) \in E$ defined as:

$$w_{i,j} = \begin{cases} -w_t & \mathcal{I}_{v_i} = \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ w_c & \mathcal{I}_{v_i} \neq \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ 0 & (v_i, v_j) \in E/E_q \end{cases} \tag{5.1}$$

where $w_c > w_t$.

With the formulation above, the planning problem can be written as an optimization problem, where we want to balance between the co-observation performance and total number of flows (cross-trajectory robots) needed. For convenience, we use (ij) to represent the edge (v_i, v_j), and $(+i)$ to represent the edge (v^+, v_i).

$$\min \quad \sum_{k}^{\mathcal{K}} \sum_{(+i)\in E} f_{+i}^k - \rho \sum_{k}^{\mathcal{K}} \sum_{(ij)\in E} w_{ij} f_{ij}^k \tag{5.2a}$$

$$\text{subject to} \quad \sum_{\{h:(hi)\in E\}} f_{hi}^k = \sum_{\{j:(ij)\in E\}} f_{ij}^k, \qquad \forall k, \forall v \in V_q/\{v^+, v^-\} \tag{5.2b}$$

$$\sum_{k}^{\mathcal{K}} \sum_{\{i:(ij)\in E\}} f_{ij}^k \geq 1 \qquad \forall v_j \in \{V_p^s\} \tag{5.2c}$$

$$f_{ij}^k \in \{0, 1\} \qquad \forall (ij) \in E \tag{5.2d}$$

The first term in 5.2a is total outgoing flow from the source v^+ while the second term is the total cost of all flows which represents the overall co-observation performance (defined as the total number of cross-trajectory edges taken by all flows beyond the regular trajectory edges). The constant ρ is a penalty parameter manually elected to balance between two terms in the cost function. The constraint (5.2b) is the flow conservation constraint, which ensures that the amount of flow entering and leaving a given node v is equal (except for v^+ and v^-). The constratin (5.2c) is the flow coverage constraints, which ensures that all security checkpoints $\{V_p^s\}$ have been visited by

at least one flow (robot). Since the security graph G_q is acyclic, this problem is in complexity class P, and can be solved in polynominal time [Ntafos and Hakimi, 1979].

5.6 Co-observation performance

Notice that problem (5.2) is guaranteed to have a solution for $\mathcal{K} = N_p$ where all $\mathcal{K}$ robots follows the reference trajectory ($f_{ij}^k = 1, \forall \mathcal{I}_{v_i} = \mathcal{I}_{v_j} = k$). Additionally, for a fixed number of robots $\mathcal{K} \geq N_p$, it is possible to have a subset of flows $F_{\mathcal{E}} \in F$ that is empty, i.e. $f_{ij} = 0, \forall (v_i, v_j) \in E$, $f \in F_{\mathcal{E}}$, these flow will not increase the cost and will not have any impact on the solution. The trade-off between the number of surveillance robots used, and the overall security performance is a critical consideration in this case. Increasing the number of robots generally contributes to improved security by enabling more cross-trajectory co-observations. However, adding more robots can lead to diminishing returns, as the benefits gained from additional robots may be counterbalanced by increased complexity and coordination challenges. To capture this trade-off, the penalty parameter ρ in the cost function 5.2a allows us to fine-tune the balance between the number of robots employed and the desired task performance. By optimizing this parameter, we can strike a suitable equilibrium between security enhancement and operational efficiency.

To find such tradeoff, we employ an iterative approach. We start with $\mathcal{K} = 1$ and gradually increase it until F contains a empty flow. This indicates that the performance can no longer be improved by adding more robots. To ensure that the value of $\mathcal{K}$ does not grow unbounded, penalty parameter ρ need to ensure that the second term for a single flow in (5.2a) $\rho \sum_{(ij)\in E} w_{ij} f_{ij}^k$ is always smaller than 1.

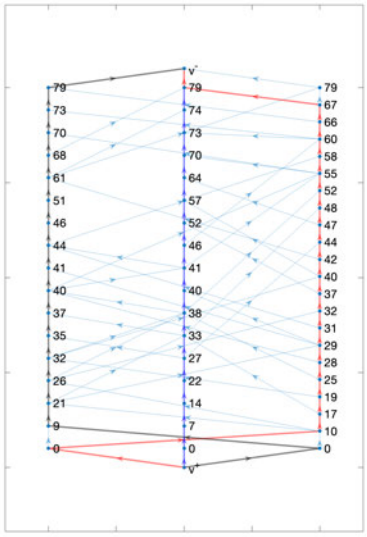


Figure 5·6: Security Graph and result for 3 flows

5.7 Result and simulation

In this section, we test the proposed method starting from the results in Figure 3·2, where three trajectories are provided for the map exploration task with no security related constraints (co-observation schedule and reachability). We use the same environment and dynamic setup, where we have a $10m \times 10m$ task space, two forbidden regions (two rectangles in 3·2) and robots with a max velocity of $0.5m/dt$.

We start by adding a total of $\mathcal{K} = 3$ surveillance robots into the workspace using the parameters $w_c = 10$, $w_t = 1$ and $\rho = 0.01$. The three trajectories are transformed into the graph G shown in Figure 5·6, where the number on each vertex v_i represents the corresponding time t_i. Vertices in each column belong to the same trajectory, and edges across different columns are cross-trajectory edges.

The flows derived from the solution of the optimization problem (5.2) are highlighted in the graph. The planning result in the workspace is shown in 5·7 as

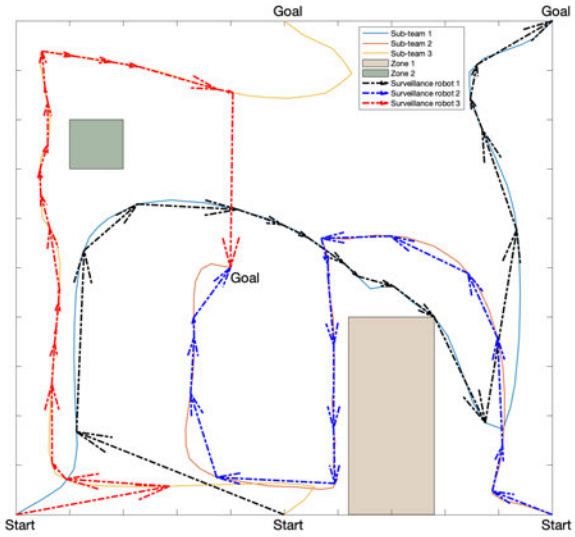


Figure 5·7: Result of 3 surveillance agents' plan in workspace

dash-dotted arrows with the same color used for each flow in 5·6. Notice that flows represent the robot co-observation plan instead of actual robots, such that the red flow, for example, shows that the sub-team 2 is expecting a robot from sub-team 1, departing at $t = 0$ and arriving at $t = 10$, and needs to send a robot (not necessarily the same one received from sub-team 1) to sub-team 3, departing at $t = 67$ and arriving at $t = 79$. This plan is not ideal in terms of our security criteria since co-observations happens between the same pair of surveillance robot and sub-team for the majority of the time with only three cross-trajectory edges that are covered.

For the case where $\mathcal{K} = 4$, the results are shown in Figure 5·8-5·9. With one additional robot added, there are significant increase in the total cross-trajectory edges covered, and sub-teams performs co-observation with different robots during the task period.

If we further increase $\mathcal{K} > 4$, we do not get a better result; instead, the planner will generate 4 flows with the rest $\mathcal{K} - 4$ flows empty.

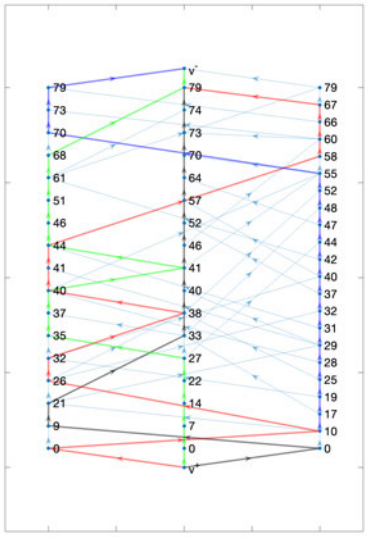


Figure 5·8: Security Graph and result for 4 flows.

5.8 Summary

In this chapter, we provide a method to enhance the security of a multi-robot system without sacrificing the performance. This is done by introducing additional robots to perform cross-trajectory co-observations by traveling between different trajectories. We model the unsecured multi-robot trajectories as a checkpoint graph by identifying checkpoints that requires observation and cross-trajectory paths that can safely access the checkpoints from different trajectories. We have shown that the co-observation planning problem across different trajectories can be transformed into a multi flow problem on the graph, and that we can find the minimal number of robot needed to finish the co-observation task.

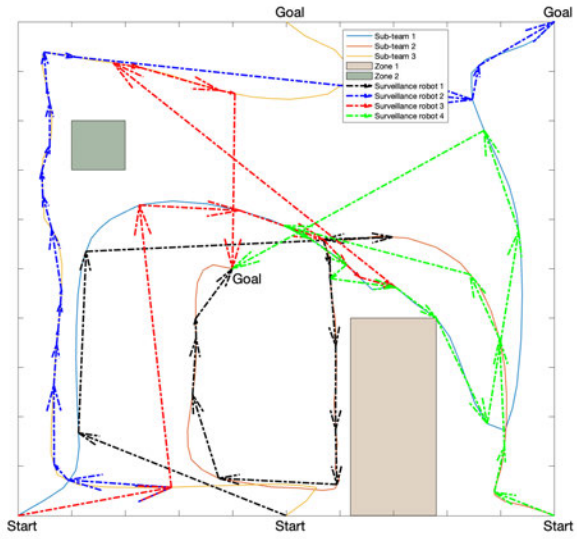


Figure 5·9: Result of 3 surveillance agents' plan in work space.

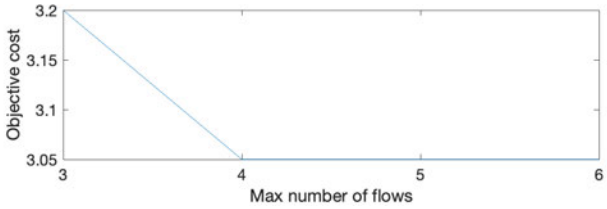


Figure 5·10: Plan reach optimality when $\mathcal{K} = 4$, further increase of $\mathcal{K}$ does not increase the cost of objectives.

Chapter 6

Distributed Online Task Assignment and Control with Applications to Security

In practical applications, there might be situations that cannot be planned for ahead of time (e.g., visiting unexpected targets of interest). These online events will unavoidably lead to conflicts with the security spatio-temporal requirements (e.g., reaching an unexpected target might lead to missing a checkpoint). A complete optimal replan in such cases might be unfeasible (due to the cost of high-dimensional computations and limited communications or security requirements).

To allow for deviations from the planned objectives while still maintaining the spatio-temporal requirements (i.e. the co-observation plan), our solution includes a distributed task assignment algorithm that assigns tasks among the sub-team to optimize the use of the additional robots. The algorithm ensures that some robots will fulfill the original plan and maintain the required safety protocols, so that others can be assigned to handle online tasks. We assume that we do not have prior information about which robots within a sub-team are more likely to be compromised. Therefore, we avoid choosing a single central agent to handle the task assignment problem; instead, we develop a distributed task assignment algorithm that can work even if not every robot can communicate with the entire team. The original security guarantees are maintained by enforcing safe regroup times through a pre-computed reachability analysis.

To summarize, the contribution of this chapter are as follows:

1) We provide a way to dispatch and compute deviation and regroup times for sub-teams using approximated reachability analysis.

2) We introduce a computationally inexpensive distributed task assignment algorithm that utilizes Alternating Direction Method of Multipliers (ADMM) and consensus-like algorithms. This algorithm can handle different priority constraints, such as guaranteeing preplanned tasks versus optional online tasks.

3) We combine the assignment algorithm with Control Barrier Functions (CBFs) for real-time control, demonstrating its applicability for security in multi-agent systems.

6.1 Definitions and Preliminaries

6.1.1 Communication graph

We consider that agents in the same sub-team p can exchange information according to an undirected communication graph G_p with Laplacian matrix L_p.

Definition 5. *Let A be the adjacency matrix of a symmetric network. Let $D = \text{diag}(A\mathbf{1})$. Then the Laplacian matrix is defined as $L = D - A$. [Chung, 1997]*

We assume that the network graph is always connected and symmetric, which implies the following:

Lemma 1. *$L\mathbf{1} = 0$ and L is* posivie semidefinite.

Proof. let $\mathbf{x} \in \texttt{null}(L)$, i.e. $L\mathbf{x} = 0$, which implies

$$\mathbf{x}^{\mathrm{T}} L\mathbf{x} = \mathbf{x}^{\mathrm{T}}(\sum_{e\in E} L)\mathbf{x} = \sum_{e\in E} \mathbf{x}^{\mathrm{T}} L\mathbf{x} = \sum_{(i.j)\in E} (x_i - x_j)^2 = 0 \tag{6.1}$$

where E is the set of edges in graph G. This implies that $L > 0$, and $x_i = x_j$ for every $(i, j) \in E$ which, for a connected graph, means that all x_i are equal. Thus, $L\mathbf{x} = x_i L\mathbf{1} = 0$, $L\mathbf{1} = 0$ [Kelner, 2007]. □

Let $\alpha = \text{stack}(\{\alpha_i\})$ be a vector of scalar states at each node.

Lemma 2. *The computation of $L\alpha$ can be done in a distributed way for any α: each agent i only needs to know α_i and exchanging a single round of communication with its neighbors N_i.*

Proof. Give the definition of L, for an individual agent, the operation can be written as $[L\alpha]_i = \sum_{\in N_i}(\alpha_j - \alpha_i)$, hence the claim. □

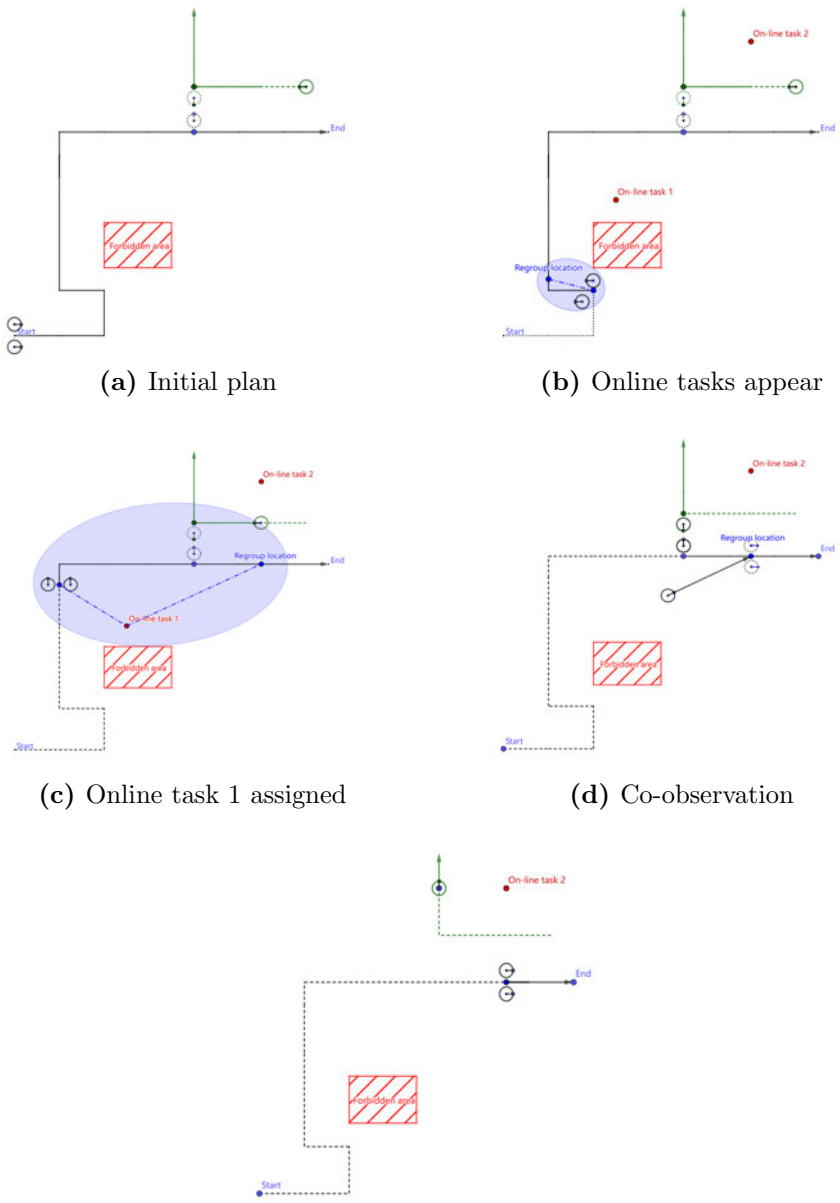
6.2 Problem overview and proposed solutions

In this section, we provide a high level overview of the task assignment problem discussing the main tasks and challenges involved. We consider two types of tasks.

Trajectory task P: This task corresponds to the original preplanned mission trajectory, with designated locations and pre-determined times for the agents to reach. The trajectory task is fulfilled by following the nominal trajectory $\{q_p\}$ given to the sub-team. These tasks have the highest priority and it is required that at least one agent in each sub-team is always assigned to this task.

Online tasks $\{O_j\}$: These are tasks that are not known at planning time. Online task j appears at a certain time t_{O_j} at a location q_{O_j} (Figure 6·1b), and it is satisfied when a robot reaches a small neighborhood $\mathcal{C}_{O_j}$ of q_{O_j}. We consider that an online task is optional, and may go unsatisfied if there are no available agents, or if the overall safety guarantees cannot be maintained with an agent deviation.

The assignment process involves several steps, as illustrated in Figure 6·1, and it is combined with the reachability ellipsoid introduced in Chapter 4. When an online task appears (Figure 6·1b), the sub-team with extra agents available will try to find a regroup time that guarantees safety during the deviation (Section 6.3). If such regroup time can be found (Figure 6·1c), the online task will be passed to the assignment algorithm (Section 6.4). The priorities in the assignment ensure that at least one agent will satisfy the trajectory task P (co-observation), while the extra



(a) Initial plan

(b) Online tasks appear

(c) Online task 1 assigned

(d) Co-observation

(e) Regroup and continue

Figure 6·1: (6·1a) Two sub-team (black and green) are assigned to follow the designed trajectory (solid lines) and perform co-observation at designated location (dotted circle). Online task 1 and 2 appear during the mission but neither is assigned. (6·1b) Green team only have one robot thus not able to fulfill online task 2. The largest current reachability region (blue ellipse), that not overlap with forbidden region, does not contain online tasks. No task is assigned. (6·1c,6·1d) A safe regroup time is found for black team and one robot is assigned to co-observation and the other one deviated for online task 1. (6·1e) Two robot are expected to see each other at regroup location to ensure a safe deviation.

robots can fulfill an online task O_j safely (Figure 6·1d). Before deviating, the robots in the sub-team must reach an agreement on a safe regroup time and location (Figure 6·1e). From a control perspective, the co-observation and regroup constraints are guaranteed through the use of time-dependent Control Barrier Functions (in Section 6.5). Note that the overall security is maintained, and none of the robots will be able to reach the forbidden regions because:

- For the robots assigned to trajectory tasks P, this type of deviation would imply breaking the constraints imposed by the co-observation schedule.
- For the robots assigned to the online tasks O_j, this type of deviation would cause them to miss the implicit co-observation constraints with the rest of the team at the regroup location and time.

In the sections below, we present our solution in three parts. Section 6.3 introduces the computation of the *regroup* time. Section 6.4 introduces the assignment of *trajectory tasks* and and *online tasks*. Section 6.5 introduces the control problem regarding the requirements by the tasks.

6.3 Regroup time Computation

In this section, we describe how to find a secured regroup time to fulfill unplanned tasks through our reachability analysis. Our main concerns for the unplanned deviation are the potential unauthorized access to certain areas. For robots departing from any part of the reference trajectory, we aim to find the latest time a robot must *regroup* with others on the reference trajectory to ensure that the reachability region during departure does not overlay any forbidden region. Similar to the checkpoint graph construction in Section 5.4.1, for a fixed waypoint on the reference trajectory, we search for another waypoint that could return the biggest reachability ellipsoid with an empty intersection with any of the forbidden regions.

For a *departure* time t_d, we find the largest *regroup* time t_r such that the reachability ellipse between the *regroup* $q(t_r)$ and *departure* $q(t_d)$ does not overlap with any of the forbidden regions, i.e.:

$$\begin{aligned} t_r = \underset{t_r}{\text{argmax}} \quad & t_r \\ \text{subject to} \quad & \mathcal{E}(q(t_d), q(t_e), t_d, t_r) \cap F = \emptyset, \end{aligned} \tag{6.2}$$

where $\{q(t)\}$ is the reference trajectory for the sub-team, and F is the union of all forbidden regions.

Instead of using the t_r as one of the reachability ellipsoid's foci in the next search like the method used in Section 5.4.1, we find the *regroup* times with respect to all possible *departure* times along the pre-defined trajectory. To reduce the online computation complexity and ensure security, the pairs $(t, t_r(t))$ for all $t \in [0, t_e]$, with t_e be the time horizon, are generated offline along with the trajectories and stored in a look-up table.

When an online task appears, agents will check for the corresponding safe regroup time (assuming they depart at that moment) in the lookup table. If the task location is not inside the corresponding reachability ellipsoid, the task will not be assigned (see, e.g. Figure 6·1b). If the reachability ellipsoid instead includes the task location, the online task will then be passed to the task assignment algorithm (see e.g. Figure 6·1c).

Since the lookup table is precomputed and stored locally on each robot, once a task is assigned, all agents (both deviating and non-deviating) will be aware of the regroup time without the need for communication. This ensures that attackers cannot modify the regroup time (unless the entire sub-team is compromised).

6.4 Distributed task assignment protocol

In this section, we describe in detail how we formulate the task assignment problem and solve it in a distributed manner using an ADMM-based approach. Inside the sub-team, each robot is either assigned to the trajectory task P, or to an online task O_j. We introduce a matrix of *assignment variables* $\alpha_{\mathcal{A}} = [\alpha_{i,j}]$, which are functions of time, and where each element $\alpha_{i,j} = 1$ represents the assignment of agent i to task j. We compute the coefficients in a distributed manner using alternating direction method of multipliers (ADMM). More formally, we define the main task assignment problem as the following Linear Program:

$$\min_{\alpha} \; -\operatorname{tr}(\boldsymbol{W}_{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\alpha}_{\mathcal{A}}) \tag{6.3a}$$

$$\text{subject to} \sum_{j} \alpha_{ij} = 1 \tag{6.3b}$$

$$\sum_{i} \alpha_{iP} = 1 \tag{6.3c}$$

$$\sum_{i} \alpha_{ij} \leq 1 \quad \text{for} \quad j \neq P \tag{6.3d}$$

$$0 \leq \alpha_{ij} \leq 1 \tag{6.3e}$$

where $\boldsymbol{W}_{\mathcal{A}}, \boldsymbol{\alpha}_{A} \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{T}|}$ are weight and assignment matrices, $j \in \mathcal{T}$, $i \in \mathcal{A}$, and $\mathcal{T} \subseteq P \cup \{O_j\}$ is the set of trajectory and online tasks while $\mathcal{A}$ is the set of agents. The rationale behind the constraints is as follows:

(6.3b) ensures that each agent is assigned to exactly one task. This is a local constraint easy to handle in the local step of ADMM.

(6.3c) ensures that the trajectory task $P \in \mathcal{T}$ is always assigned exactly once.

(6.3d) ensures that online tasks do not get assigned to more than one agent. When $|\mathcal{A}| = |\mathcal{T}|$, this constraint can be written as $\sum_i a_{ij} = 1$

Each entry $\boldsymbol{W}_{\mathcal{A}} = [w_{ij}]$ represents the weight for task j and agent i, and is determined as follows:

Trajectory task P : $w_{iP}(x_i) = (d(\bar{q}, x_i) + \epsilon)^{-1}$

Online tasks $\{O1, O2, \ldots\}$: $\{O1, O2, \ldots\}$, $w_{iO_m}(x_i) = (d(q_{O_m}, x_i) + \epsilon)^{-1}$

where x_i is the state for robot i, and ϵ is a small constant. In both cases, $w_{ij} \leq \epsilon^{-1}$ (the reason behind this choice became clean when discussing practices below). Notice that, we use term x_i instead of the reference trajectory q to represent the actual state of a robot for real-time control purpose.

We first show a distributed solution for problem (6.3) when we have an equal number of agents and tasks, i.e. $|\mathcal{A}| = |\mathcal{T}|$; we extended the solution for other cases in later sections.

6.4.1 Distributed optimization problem formulation

When the number of agents and targets (trajectory and online) is the same, i.e., $|\mathcal{A}| = |\mathcal{T}|$, the problem can be written in matrix form (dropping the subscript $\mathcal{A}$ to simplify the notation) as:

$$\min_{0 \leq \alpha \leq 1} - \sum_i (\mathbf{w}_i^{\mathrm{T}} \alpha_i) \tag{6.4a}$$

$$\text{subject to } \boldsymbol{\alpha}\mathbf{1} = \mathbf{1} \tag{6.4b}$$

$$\mathbf{1}^{\mathrm{T}} \underbrace{\boldsymbol{\alpha}_{\mathbf{e}_j}}_{\boldsymbol{\alpha}_j} = 1 \quad \forall j \in \mathcal{T} \tag{6.4c}$$

where $\boldsymbol{\alpha}_j \doteq \boldsymbol{\alpha}_{\mathbf{e}_j}$ is the j-th columns of $\boldsymbol{\alpha}$, and $\boldsymbol{\alpha}_i$ is the i-th row of $\boldsymbol{\alpha}$.

Our strategy is to rewrite the problem in a separable form and introduce distributed computations for parts that involve constraints over all the agents. Following the

ADMM framework [Boyd et al., 2011], we separate the problem in two parts:

$$\min_{\mathrm{dom} f} \sum_i f_i(\alpha_i) + g(\mathbf{z}) \tag{6.5a}$$

$$\text{subject to } \boldsymbol{\alpha} = \mathbf{z} \tag{6.5b}$$

where

$$f_i(\boldsymbol{\alpha}_i) = -\mathbf{w}_i^{\mathrm{T}} \boldsymbol{\alpha}_i \tag{6.6}$$

$$\mathrm{dom} f = \{\boldsymbol{\alpha}_i | \alpha_i^{\mathrm{T}} \mathbf{1} = 1, 0 \leq \boldsymbol{\alpha} \leq 1\} \tag{6.7}$$

$$g(\mathbf{z}) = \begin{cases} 0 & \text{if } \mathbf{z}^{\mathrm{T}} \vec{1} = \vec{1} \\ -\inf & \text{otherwise.} \end{cases} \tag{6.8}$$

The Lagrangian of the problem can be written as:

$$\mathcal{L}(\boldsymbol{\alpha}, \mathbf{z}, \mathbf{u}) = \sum_i f_i(\boldsymbol{\alpha}_i) + g(\mathbf{z}) + (\rho/2)\|\boldsymbol{\alpha} - \mathbf{z} + \mathbf{u}\|_2^2 \tag{6.9}$$

We want to separate the non-distributed (6.3b) and distributed (6.3c,6.3d) constraints in (6.3), by rewriting them in the form of (6.7,6.8). The local constraint (6.7), the replacement of (6.3b), enforces that each agent is assigned exactly to a single task in $\mathcal{T}$. The distributed constraint (6.8), the replacement of (6.3c,6.3d), enforces that each task is assigned exactly to a single agent.

The ADMM framework then proceeds with iterations through the following steps:

1) α-update:

$$\begin{aligned} \boldsymbol{\alpha} \leftarrow \underset{\alpha}{\mathrm{argmin}} & -\sum_{i=1}^{n} \mathbf{w}_i \boldsymbol{\alpha}_i^{\mathrm{T}} + (\rho/2)\|\boldsymbol{\alpha} - \mathbf{z} + \mathbf{u}\|_2^2 \\ \text{subject to } & \boldsymbol{\alpha}^{\mathrm{T}} \vec{1} = 1 \\ & \vec{0} \leq \boldsymbol{\alpha} \leq \vec{1} \end{aligned} \tag{6.10}$$

The optimization problem is separable, and the local objective for agent i can

be written as a QP problem:

$$\begin{aligned} \min_{\alpha_i} \quad & \frac{1}{2}\boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{\alpha}_i + (-\mathbf{z}_i^k + \mathbf{u}_i^k - \frac{\mathbf{w}_i}{\rho})^{\mathrm{T}}\boldsymbol{\alpha}_i \\ \text{subject to} \quad & \mathbf{1}^{\mathrm{T}}\alpha_i = 1 \\ & \vec{0} \leq \boldsymbol{\alpha}_i \leq \vec{1}. \end{aligned} \tag{6.11}$$

Each agent can compute its part of the solution independently.

2) **z**-update:

$$\mathbf{z} \leftarrow \underset{\mathbf{z}^{\mathrm{T}}\mathbf{1}=\mathbf{1}}{\operatorname{argmin}} \sum_{j=1}^{m} \frac{1}{2}\|\mathbf{z} - (\alpha^{k+1} + \mathbf{u}^k)\|^2 \tag{6.12}$$

This problem is separable over $\mathbf{z}_j$, the columns of z, i.e., over tasks. Each sub-problem (i.e., the solution for task j) can be written as:

$$\min \frac{1}{2}\|\mathbf{z}_j - \alpha_j^{k+1} - \mathbf{u}_j^k\|^2 \tag{6.13a}$$

$$\text{subject to } \mathbf{1}^{\mathrm{T}}\mathbf{z}_j = 1 \tag{6.13b}$$

where $\mathbf{z}_j$, $\boldsymbol{\alpha}_j$ and $\mathbf{u}_j$ are all jth column of the corresponding matrices. The objective can be written as $\sum_i (\mathbf{z}_i - \alpha_i - \mathbf{u}_i)^2$, which is separable over agents, but the constraint is not, so we cannot directly solve this step in a distributed way. However, we show that this can be solved using distributed *projected gradient descent* in Section 6.4.2.

3) **u**-update:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i^k + \boldsymbol{\alpha}_i^{k+1} - \mathbf{z}_i^{k+1} \tag{6.14}$$

Each multiplier **u** updates only need information locally to the node, so this step is distributed and can be executed independently at each node.

6.4.2 Distributed z-update

When the communication graph G_p is connected, we show that problem (6.13) can be solved using a distributed projected gradient descent, where each robot requires only information from their neighbours.

Lemma 3 (Projected gradient descent)**.** *Let Ω be a convex set of constraints on a state vector y and let f be a convex function that we want to minimize. Then, the iteration $y[k+1] = \mathrm{proj}_\Omega(y[k] - \epsilon\Gamma \operatorname{grad}_y f(y[k]))$ converges to the constrained minimum for any positive definite matrix $\Gamma \succ 0$.*

See [Bertsekas and Tsitsiklis, 2000] for a proof.

Lemma 4. *The iterations $y[k+1] = y[k] + Lv[k]$ satisfy $\mathbf{1}y[k] = \mathbf{1}^{\mathrm{T}}y[0]$ for all k and any v.*

Proof. Using the properties of the Laplacian matrix in Lemma 1, we have the following invariant: $\mathbf{1}^{\mathrm{T}}y[k+1] = \mathbf{1}^{\mathrm{T}}y[k] + \mathbf{1}^{\mathrm{T}}Lv[k] = \mathbf{1}^{\mathrm{T}}y[k]$, from which the claim follows. □

The *gradient* of problem (6.13) can be computed in a distributed way: $\operatorname{grad}_{\mathbf{z}_j} \mathcal{L} = (\mathbf{z}_j^k - \boldsymbol{\alpha}_j^{k+1} - \mathbf{u}_j^k)$.

Thus by iterating

$$\begin{aligned} \mathbf{z}_j[k'+1] &= \mathbf{z}_j[k'] - \varepsilon L \operatorname{grad}_{\mathbf{z}_j} \mathcal{L} \\ &= \mathbf{z}_j[k'] - \varepsilon L(\mathbf{z}_j[k'] - \alpha_j^{k+1} - \mathbf{u}_j^k) \\ &= (I - \varepsilon L)\mathbf{z}_j[k'] + \varepsilon L(\alpha_j^{k+1} + \mathbf{u}_j^k) \end{aligned} \tag{6.15}$$

we show that

- $\mathbf{z}[k']$ converges to the desired solution of (6.12)
- Each step is distributed.

Note that for a state vector v,

$$L\mathbf{v} = (D - A)\mathbf{v} = D\mathbf{v} - A\mathbf{v}, \tag{6.16}$$

which, if we consider only the i-th entry, becomes

$$d_i v_i - \sum_{j:(i,j)\in E} a_{ij} v_j, \tag{6.17}$$

where $d_i = \sum_{j:(i,j)\in E} a_{ij}$. Using the above, we can write 6.15 for each agent as

$$\mathbf{z}_i[k'+1] = \mathbf{z}_i[k'] + \varepsilon \sum_{(i,j)\in E} \big(\mathbf{z}_i[k'] - 2(\mathbf{z}_j[k'] - \boldsymbol{\alpha}_j[k] - \mathbf{u}_j[k])\big) \tag{6.18}$$

Note the different time index for $\boldsymbol{\alpha}_j$ and $\mathbf{u}_j$ (k instead of k'): this is because $\boldsymbol{\alpha}_i$ and $\mathbf{u}_i$ are constant while we update the $\mathbf{z}_i$'s. In practice, rather than waiting for the convergence of the projected gradient desent, we only run a few iterations of (6.15) in each ADMM iteration (5 iterations in the simulations of Section 6.6). Although (6.13a) decreases monotonically with each iteration, the impact of an inexact solution on the ADMM will not be discussed here, and its analysis is left as future work.

6.4.3 Shadow agents and secondary trajectory tasks

The solution of problem (6.4) is based on the idea that one task is assigned to *exactly* one agent. This cannot be the case when the number of robots and the number of tasks are not equal, i.e. $|\mathcal{A}| \neq |\mathcal{T}|$. For these cases, we introduce the concepts of *secondary trajectory task* and *shadow agent*, transforming formulation (6.3) in the same form as (6.4) for which we can then apply the same algorithm.

Secondary trajectory tasks P' ($|\mathcal{A}| > |\mathcal{T}|$)

These tasks are introduced when we have more agents than tasks (i.e. $|\mathcal{A}| > |\mathcal{T}|$). The secondary trajectory task captures the marginal gains of having more than one agent following the planned trajectory. The weight for P' should be lower than every weight for online tasks and the trajectory task P; we propose $w_{iP'}(x_i) = (d(P, x_i) + \epsilon')^{-1}$ with $\epsilon' \gg \epsilon$. In this way, task P' will be assigned only if task P and $\{O_j\}$ have all

been assigned (see Proposition 5 below).

Proposition 5. *Agents in $\mathcal{A}$ will be assigned to P' only if all tasks in P and $\{O_j\}$ have been already assigned to agents in $\mathcal{A}$.*

Proof. For $|\mathcal{A}| > |\mathcal{T}|$, assume that for the original problem (6.3), it exists an optimal solution of $\alpha \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{T}|}$. We can always find an arbitrary robot A_1 get assigned to task $T_i \in \mathcal{T}$, and robot A_2 got no task assigned to. In such case, $w_{T_i}(A_1) > w_{T_i}(A_2)$. By way of contradiction, assume there exists a different optimal value $\boldsymbol{\alpha}' \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{A}|}$ for the modified problem (6.4) that assign A_1 with secondary trajectory task P' and A_2 with T_i. We will have

$$(w_{P'}(A_1) + w_{T_i}(A_2)) - (w_{T_i}(A_1) + w_{P'}(A_2)) > 0 \tag{6.19}$$

Since $\epsilon' \gg \epsilon$, we have $w_{T_i}(A_2)) \gg w_{P'}(A_1)$ and $w_{T_i}(A_1) \gg w_{P'}(A_2)$. Then, (6.19) results $w_{T_i}(A_2) > w_{T_i}(A_1)$ which contradict with the assumption. □

Shadow agent ($|\mathcal{A}| < |\mathcal{T}|$)

For cases where the number of tasks is larger than the number of agents, we introduce a set of *shadow* agents $\mathcal{A}_S = \bigcup_i \mathcal{A}_{S,i}$, where $|\mathcal{A}_S| = |\mathcal{T}| - |\mathcal{A}|$. Shadow agents $\mathcal{A}_{S,i}$ are virtual replicas of an agents i, and are considered to have the same location and graph connectivity as agent i (in fact, communications and computations are handled by the physical agent i). Meanwhile, we consider that all the agents in $\mathcal{A}_{S,i}$ can always communicate with physical agent i, and vice versa. Finally, if physical agent $i \in \mathcal{A}$ is connected with physical agent $j \in \mathcal{A}$, then all the agents in $\mathcal{A}_{S,i}$ can communicate with all the agents in $\mathcal{A}_{S,j}$. There might be more than one shadow agent for a single physical agent, and the partition of shadow agents across physical agents can be arbitrary.

Assigning tasks to shadow agents means that the corresponding tasks are abandoned, and will not be performed by an actual agent. We define the weights of the shadow agents as

Trajectory task P : $w_{S_iP}(x_{S_i}) = -M$, where $M > 2\epsilon^{-1}$. Effectively, this implies that it is never advantageous, from the optimization perspective, to assign a shadow agent to the trajectory task (see Proposition 6 below).

Online tasks$\{Oj\}$ $w_{S_iO_j}(x_{S_i}) = -(d(O_j, x_{S_i})+\epsilon)^{-1}$ This implies that assigning tasks to shadow agents (i.e., abandoning the tasks), from the optimization perspective, is never encouraged; moreover, farther tasks are more likely to be abandoned.

We now show that solving the modified problem with *Secondary trajectory task* or *shadow agent* is equivalent to solving it without.

Proposition 6. *Agents in $\mathcal{A}_G$ will never be assigned to P when $M > 2\epsilon^{-1}$*

Proof. From the constraint, only one agent a (among regular agents $\mathcal{A}$ or shadow agents $\mathcal{A}_G$) will be assigned to P; we want to show that $a \in \mathcal{A}$. By way of contradiction, assume α to be a solution that is feasible (i.e., satisfies the constraints) and optimal, but with $a \in \mathcal{A}_G$; we have that the associated weight is $w_a = -M$; take the real agent $a' \in \mathcal{A}$ corresponding to a; by construction, a' will be assigned to one of the online tasks, with weight $w_{a'} = (d(O, x_a) + \epsilon)^{-1}$, $0 < w_{a'} \leq \epsilon^{-1}$. Let w_p be the weight of a' if it was assigned to P. Now consider α', which is the same as α but with a and a' swapped. Then we have

$$\begin{aligned}\mathrm{tr}(W^{\mathrm{T}}\boldsymbol{\alpha}') - \mathrm{tr}(W^{\mathrm{T}}\alpha) &= (w_p - w_{a'}) - (-M + w_{a'}) \\ &= (M + w_p) - 2w_{a'} \geq M - 2\epsilon^{-1} \quad (6.20)\end{aligned}$$

By assumption, $M > 2\epsilon^{-1}$, hence $\mathrm{tr}(W^{\mathrm{T}}\boldsymbol{\alpha}') - \mathrm{tr}(W^{\mathrm{T}}\alpha) > 0$, hence α cannot be optimal, leading to a contradiction. □

6.5 CBF-based continuous time control

6.5.1 Control Barrier Function

We first review the definition of Control Barrier Functions [Xu et al., 2015]. Consider an affine-input dynamical system

$$\dot{x} = f(x) + g(x)u \tag{6.21}$$

with f and g locally Lipschitz continuous, $x \in \mathbb{R}^n$ and $u \in U \subset \mathbb{R}^n$. To simplify our discussion, we consider the case where x_i represents the position of the i-th robot (although, as already mentioned, the state could contain higher-order quantities, such as velocity).

Given a set $\mathcal{C} \subset \mathbb{R}^n$ defined as $\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}$ for a time-varying continuously differentiable function $h(x,t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$. The function h is called a zeroing control barrier function, if there exists a class $\mathcal{K}$ function γ such that

$$\sup_{u \in U}[L_f h(x,t) + L_g h(x,t)u + \frac{\partial h(x,t)}{\partial t} + \gamma(h(x,t))] \geq 0 \tag{6.22}$$

where $L_f h(x,t) = \frac{\partial h(x,t)}{\partial x} f(x)$ [Lindemann and Dimarogonas, 2018].

6.5.2 Control calculation

In this section, we use CBF constraints to generate online control outputs based on the assignment result and the regroup constraints. As introduced in Section 6.2, both a *trajectory task* P and an *online task* O_j result in a deadline time and location, meaning that a robot needs to reach a *co-observation* (for the trajectory task) and a *regroup* (for the online tasks) location before the assigned time.

For each robot i, we capture these deadlines with a function $h_{\mathcal{C}_m}(x_i, t)$ that contains the difference between the deadline time and the shortest possible time in which an agent can reach $\mathcal{C}_m$ from location x_i at time t. Specially, assuming a first-order

integrator dynamics, we define

$$h_{\mathcal{C}_m}(x_i, t) = (t_m - t) - \frac{d_{\mathcal{C}_m}(x_i(t))}{v_{max}}, \tag{6.23}$$

where v_{max} is the maximum speed of robot i, and $d_{\mathcal{C}_m}$ represents the distance between x and the boundary of the task region $\mathcal{C}_m$. $\{x \in \mathbb{R}^n : h_{\mathcal{C}_m}(x,\, t) \geq 0\}$ represents the set of states at time t from which the robots can potentially satisfy the deadline associated to $\mathcal{C}_m$, i.e. the safety set for robot i.

In our application, the deadline location for the *trajectory task* P (co-observation location) and *online tasks* O(regroup location) are on the pre-defined trajectory $\bar{x}_i$ at a determined time t_m, thus $d_{\mathcal{C}_m}(x_i(t)) = \|x_i(t) - \bar{x}_i(t_m)\|$.

We define $\mathbf{h}_{\mathcal{C}}(x_i(t)) = [h_{\mathbf{C}_{iP}}(x_i(t)), h_{\mathbf{C}_{iO_1}}(x_i(t)), \dots]$ as deadline barrier functions for each available task, and $J_i(u_i, x_i) = [J_{iP}, J_{iO_1}, \dots]$ as the cost function designed for each available task. The objectives for trajectory task J_{iP} will keep the robots along the reference trajectory, and J_{iO_j} will drive robots toward the online task location fast enough such that the following deadline constraints can be met.

We then propose to solve the following local constrained optimization problem at every time t to get the optimal control law $u_i(t)$.

$$\min_{u_i \in U} \boldsymbol{\alpha}_i^{\mathrm{T}} J_i(u_i, x_i) \tag{6.24a}$$

$$\text{subject to } \boldsymbol{\alpha}_i^{\mathrm{T}}(\dot{\mathbf{h}}_{\mathcal{C},i} + \gamma(\mathbf{h}_{\mathcal{C},i})) \geq 0 \tag{6.24b}$$

$$\dot{h}_{\mathcal{D},i} + \gamma(h_{\mathcal{D},i}) \geq 0 \tag{6.24c}$$

$$\dot{x}_i = f(x_i) + g(x_i)u_i \tag{6.24d}$$

where $\boldsymbol{\alpha}_i$ is the task assignment result from Section 6.4, and $h_{\mathcal{D},i} = d_{\mathcal{D}}(x_i)$ is the barrier function representing the distance between agents and obstacles.

6.6 Result and simulation

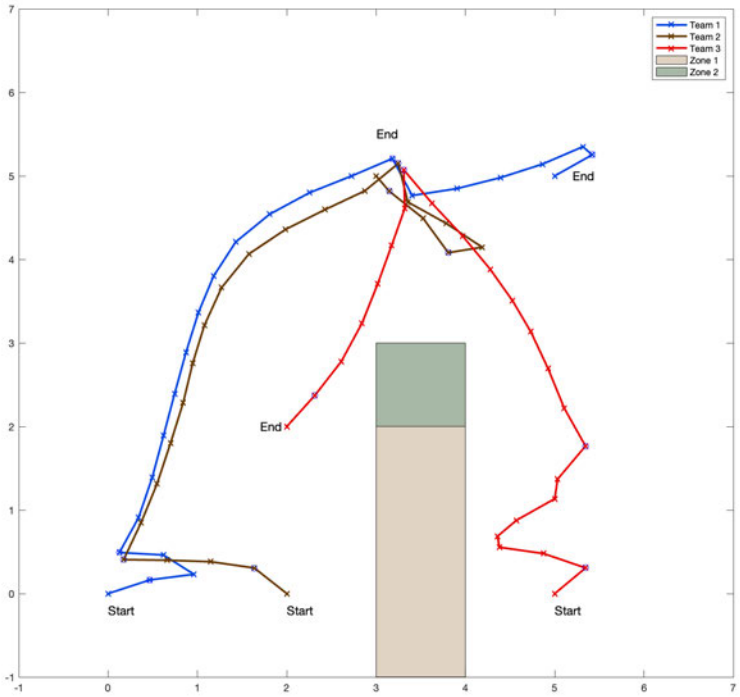


Figure 6·2: Pre-generated reference trajectory for three sub-teams.

In this section, the proposed strategy is tested using the map exploration secured with a co-observation schedule for three sub-teams. With the same setup of Section 4.5, we assume that the environment is a square region with the coordinate origin $(0, 0)$ located at the bottom left corner and the edge length of 8 units. In the test area, *zone 1* is an obstacle and, *zone 2* is a forbidden region. All robots are set to have a maximum velocity of $0.5m/s$ with a total task time of $20s$. Results in Figure 4·7 is used as the reference trajectory, which comes with a co-observation plan: team 1 and 2 meet at time 8 and 14, while team 2 and 3 meet at time 18. The reference trajectory and the environment setup are shown in Fig.6·2.

In the real-time simulation, group 1 has three robots, while group 2 and 3 have one. We will focus on the task assignment performance and deadline setup for group 1.

We assume total four *online* tasks appeared during the whole process. At time $t = 4$, task 1 appears at $(-1, 3)$ and task 4 appears at $(6, 3)$. At time $t = 15$, task 2 appears at $(2, 6.5)$ and task 3 appears at $(0, 6)$.

When the mission first begins, there are no *online* tasks, and all agents perform the trajectory task. When *online* task 2 appears and a secured regroup time was found at $t = 24$ (see e.g. Figure.6·3), and agent 1 is assigned to *online* task 1. Hence, agent 1 sets up a deadline Control Barrier Function to guarantee the return to reference trajectory before the *regroup* time.

When *online* tasks 1 and 3 appear, and are sent to the assignment algorithm later (as shown in Fig.6·4), *online* task 1 is assigned to agent 2, while task 3 is assigned to a shadow agent due to an insufficient number of agents (agent 3 was assigned to a *trajectory* task and agent 1 was still working on getting back to the reference trajectory). Online task 4 is never sent to the assignment process because it is always outside the secure reachability regions.

During the entire task process, two out of four *online* tasks appeared got fulfilled. Meanwhile, agent 3 strictly follows the trajectory and the co-observation schedule to vouch for the security of the team. The final result is shown in Fig.6·5. Both the task and co-observation had been satisfied, and when agents separate, they successfully return to the reference trajectory before the *regroup* deadline.

6.7 Summary

In this chapter, we have presented a comprehensive solution to the multi-agent system to safely handle unplanned *online* tasks and provide a real-time simulation of the

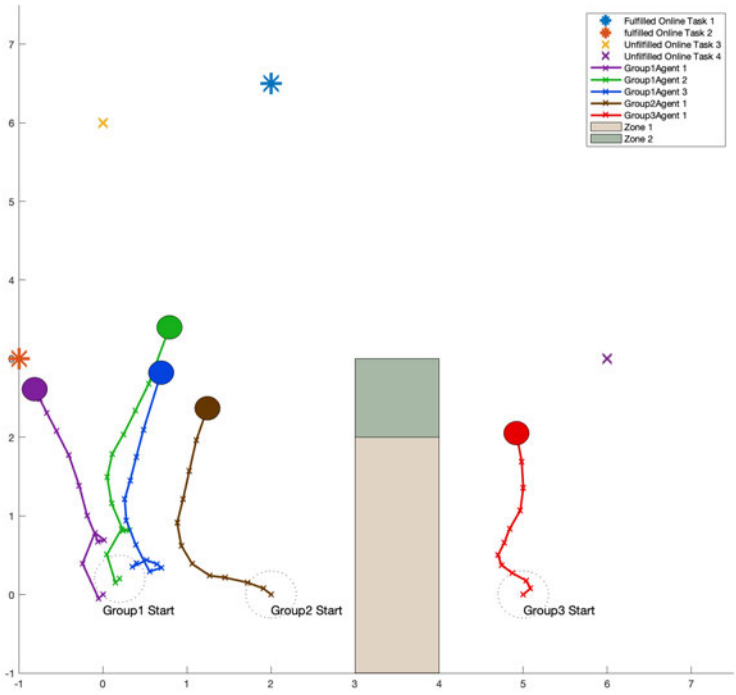


Figure 6·3: Time $t = 12$. Agent 1 has been assigned to online task 1 while agent 2 and 3 follows the trajectory

method. Our proposed method includes a security analysis that determines a secured regroup time, a distributed task assignment that can handle tasks with different priorities, and a CBF-based continuous time control to ensure spatio-temporal security requirements. Through simulation of a co-observation-scheduled map exploration task, we have demonstrated the effectiveness of our approach in efficiently handling online tasks while meeting the scheduled requirements.

In the future, we plan to investigate the impact of inexact z-update to the ADMM task assignment process, as well as the effects of network topology change during the task period. For instance, the team's composition may change due to robots joining other sub-teams or moving out of communication range while completing online tasks.

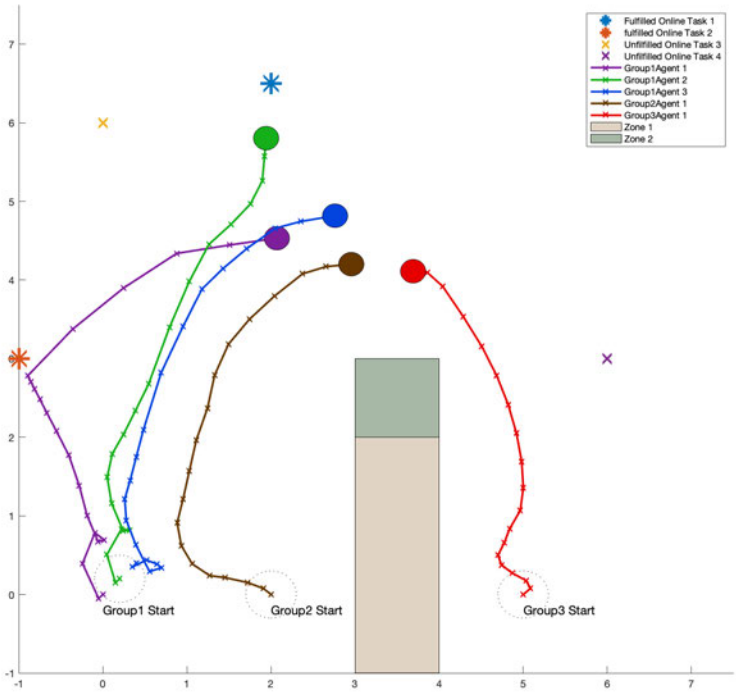


Figure 6·4: Time $t = 20$. Agent 1 gets back to the trajectory and agent 2 has been assigned to online task 2.

We will further study the impact of these changes on our approach to ensure that it remains robust and effective under various scenarios.

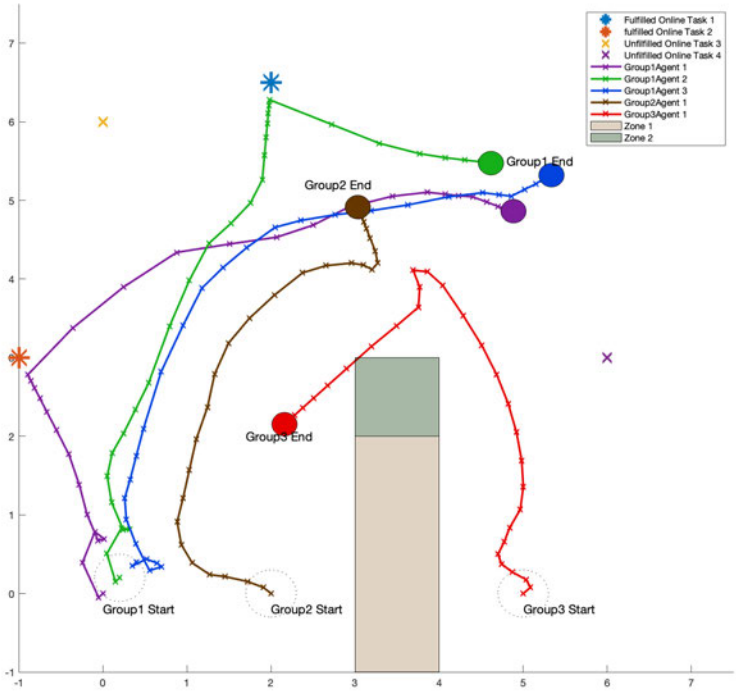


Figure 6·5: Runtime simulation with online tasks appear during the mission, online task 1 got fulfilled by agent 2 in group 1, online task 2 got fulfilled by agent 1 group 1, online task 3 and 4 is not fulfilled.

Chapter 7

Conclusion

In conclusion, this dissertation presents a novel solution based on an inter-robot observation mechanism to enhance the security of multi-robot systems against plan-deviation attacks. The proposed solution addresses several key challenges in achieving secure and efficient system operation.

Firstly, a method is proposed to transform the discrete grid-world representation of MRS trajectories into a continuous workspace. This involves the development of an ADMM-based optimal trajectory algorithm capable of handling spatio-temporal constraints and arbitrary smooth cost functions. Additionally, the reachability analysis is adapted to the continuous workspace to further enhance security in conjunction with the co-observation schedule.

Secondly, the solution incorporates the use of additional robots within each trajectory to form sub-teams. A new cross-trajectory co-observation schedule planning algorithm is introduced to facilitate the exchange of robots between sub-teams. This approach ensures the preservation of system security without compromising overall task performance.

Furthermore, to effectively handle unplanned events in real-time while maintaining the required security protocols, a distributed task assignment algorithm is proposed. This algorithm intelligently assigns tasks with different priorities, optimizing the utilization of additional robots within the system. Additionally, a CBF-based control mechanism is introduced to ensure adherence to spatio-temporal security requirements.

The combination of these techniques enables the development of secure multi-robot systems suitable for a wide range of emerging applications.

7.1 Potential future researches

In this section, we discuss some limitations of our work that could serve as avenues for future research.

The ADMM optimal solver presented in Chapter 3 is able to handles large-scale optimization problems with spatio-temporal constraints. However, one limitation is the relatively long computation time required by the solver. Future work can focus on developing distributed approaches to solve the trajectory optimization problem for each robot, as well as addressing the challenges associated with constraints affecting multiple robots in a distributed manner.

In Chapter 4, we incorporated reachability regions while considering a fixed co-observation schedule. However, the inability to adapt the focus of the ellipsoids limits the potential for achieving more secure and higher-performing results. A promising direction for future work is to develop algorithms that can dynamically optimize the co-observation schedule while satisfying the reachability requirements.

The construction of the checkpoint graph in Chapter 5 relied on a heuristic approach. Future research efforts could focus on minimizing the number of checkpoints required along each route, aiming to achieve more efficient and streamlined trajectories. Additionally, further investigation is needed to understand the relationship between variables such as w_t, w_c, and ρ, as well as their impact on the results of the multi-flow problem.

Lastly, in Chapter 6, it would be valuable to explore the effects of inexact z-updates in the ADMM task assignment process. Additionally, studying the impact of network topology changes during the task period, particularly when affected by cross-trajectory

co-observations, is an important area for future research. These open problems provide interesting opportunities for further investigation in the field.

References

Alighanbari, M. and How, J. P. (2005). Decentralized task assignment for unmanned aerial vehicles. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 5668–5673. IEEE.

Anderson, B. D. and Moore, J. B. (2012). *Optimal filtering*. Courier Corporation.

Bento, J., Derbinsky, N., Alonso-Mora, J., and Yedidia, J. S. (2013). A message-passing algorithm for multi-agent trajectory planning. In *Advances in neural information processing systems*, pages 521–529.

Bertsekas, D. P. and Tsitsiklis, J. N. (2000). Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122. `https://doi.org/10.1561/2200000016`.

Brunner, M., Hofinger, H., Krauß, C., Roblee, C., Schoo, P., and Todt, S. (2010). Infiltrating critical infrastructures with next-generation attacks. *Fraunhofer Institute for Secure Information Technology (SIT), Munich*.

Cao, Y., Yu, W., Ren, W., and Chen, G. (2012). An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438.

Choi, H.-L., Brunet, L., and How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926.

Choset, H. M., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Chung, F. R. (1997). *Spectral graph theory*, volume 92. American Mathematical Soc.

Coltin, B. and Veloso, M. (2010). Mobile robot task allocation in hybrid wireless sensor networks. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2932–2937. IEEE.

Dionne, D. and Rabbath, C. A. (2007). Multi-uav decentralized task allocation with intermittent communications: The dtc algorithm. In *2007 American Control Conference*, pages 5406–5411. IEEE.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *International Journal of Robotics Research*, 29(5):485–501.

Eberly, D. (2013). Distance from a point to an ellipse, an ellipsoid, or a hyper-ellipsoid. Geometric Tools, `https://www.geometrictools.com/Documentation/DistancePointEllipseEllipsoid.pdf`.

Forrest, C. (2017). Robot kills worker on assembly line, raising concerns about human-robot collaboration. `https://tinyurl.com/y2tzg4te`.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE.

Geramifard, A., Chubak, P., and Bulitko, V. (2006). Biased cost pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 2, pages 112–114. `https://doi.org/10.1609/aiide.v2i1.18756`.

Gil, S., Kumar, S., Mazumder, M., Katabi, D., and Rus, D. (2017). Guaranteeing spoof-resilient multi-robot networks. *Autonomous Robots*, 41:1383–1400.

Hauer, F. and Tsiotras, P. (2017). Deformable rapidly-exploring random trees. In *Robotics: Science and Systems*. `https://www.roboticsproceedings.org/rss13/p08.pdf`.

Householder, A. S. (1958). Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342.

Jin, Y., Minai, A. A., and Polycarpou, M. M. (2003). Cooperative real-time search and task allocation in uav teams. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 7–12. IEEE.

Julian, B. J., Angermann, M., Schwager, M., and Rus, D. (2012). Distributed robotic sensor networks: An information-theoretic approach. *The International Journal of Robotics Research*, 31(10):1134–1154.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2).

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*. 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 1478-1483, doi: 10.1109/ICRA.2011.5980479.

Kelner, J. (2007). Properties of the laplacian, positive semidefinite matricies, spectra of common graphs, connection to the continuous laplacian. Lecture note, Massachusetts Institute of Technology. `https://ocw.mit.edu/courses/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/resources/mit18_409f09_scribe2/`.

Khamis, A., Hussein, A., and Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*. vol 604, pp. 31–51. Springer, Cham. `https://doi.org/10.1007/978-3-319-18299-5_2`.

Lan, X. and Schwager, M. (2016). Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots. *IEEE Transactions on Robotics*, 32(5):1230–1244.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Lindemann, L. and Dimarogonas, D. V. (2018). Control barrier functions for signal temporal logic tasks. *IEEE control systems letters*, 3:96–101.

Liu, C. and Kroll, A. (2012). A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012, Zakopane, Poland, April 29-May 3, 2012, Proceedings, Part II 11*, pages 466–474. Springer.

Liu, X. and Lu, P. (2014). Solving nonconvex optimal control problems by convex optimization. *Journal of Guidance, Control, and Dynamics*, 37(3):750–765.

Mellinger, D., Kushleyev, A., and Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE international conference on robotics and automation*, pages 477–483. IEEE.

Ntafos, S. C. and Hakimi, S. L. (1979). On path cover problems in digraphs and applications to program testing. *IEEE Transactions on Software Engineering*, (5):520–529.

Pajares, G. (2015). Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs). *Photogrammetric Engineering & Remote Sensing*, 81(4):281–330.

Quinton, F., Grand, C., and Lesire, C. (2023). Market approaches to the multi-robot task allocation problem: a survey. *Journal of Intelligent & Robotic Systems*, 107(2):29.

Sariel, S. and Balch, T. (2005). Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*, pages 27–33. sn.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, 33(9):1251–1270.

Schwager, M., Julian, B. J., Angermann, M., and Rus, D. (2011). Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*, 99(9):1541–1561.

Standley, T. S. (2010). Finding optimal solutions to cooperative pathfinding problems. In *In Proceedings of the AAAI Conference on Artificial Intelligence, 24(1), 173-178.* `https://doi.org/10.1609/aaai.v24i1.7564`.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al. (2019). Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, pages 151–158.

Sujit, P. and Beard, R. (2007). Distributed sequential auctions for multiple uav task allocation. In *2007 American Control Conference*, pages 3955–3960. IEEE.

Tron, R. and Daniilidis, K. (2014). Technical report on optimization-based bearing-only visual homing with applications to a 2-d unicycle model.

Van Parys, R. and Pipeleers, G. (2016). Online distributed motion planning for multi-vehicle systems. *2016 European Control Conference, ECC 2016*, pages 1580–1585.

Walsh, W. E. and Wellman, M. P. (1998). A market protocol for decentralized task allocation. In *Proceedings International Conference on Multi Agent Systems (Cat. No. 98EX160)*, pages 325–332. IEEE.

Wang, Y., Yin, W., and Zeng, J. (2019). Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63.

Wardega, K., Tron, R., and Li, W. (2019). Resilience of multi-robot systems to physical masquerade attacks. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 120–125. IEEE.

Wardega, K., von Hippel, M., Tron, R., Nita-Rotaru, C., and Li, W. (2023). Hola robots: Mitigating plan-deviation attacks in multi-robot systems with co-observations and horizon-limiting announcements. *arXiv preprint arXiv:2301.10704*.

Xu, X., Tabuada, P., Grizzle, J. W., and Ames, A. D. (2015). Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61.

Yu, J. and LaValle, S. M. (2013). Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*, pages 157–173. Springer.

Zhu, M. and Martinez, S. (2013). On the performance analysis of resilient networked control systems under replay attacks. *IEEE Transactions on Automatic Control*, 59(3):804–808.

CURRICULUM VITAE