1-1-2014

Safe, Scalable, and Complete Motion Planning of Large Teams of Interchangeable Robots

Matthew Turpin
University of Pennsylvania, mturpin@seas.upenn.edu

Safe, Scalable, and Complete Motion Planning of Large Teams of Interchangeable Robots

Abstract
Large teams of mobile robots have an unprecedented potential to assist humans in a number of roles ranging from humanitarian efforts to e-commerce order fulfillment. Utilizing a team of robots provides an inherent parallelism in computation and task completion while providing redundancy to isolated robot failures. Whether a mission requires all robots to stay close to each other in a formation, navigate to a preselected set of goal locations, or to actively try to spread out to gain as much information as possible, the team must be able to successfully navigate the robots to desired locations.

While there is a rich literature on motion planning for teams of robots, the problem is sufficiently challenging that in general all methods trade off one of the following properties: completeness, computational scalability, safety, or optimality. This dissertation proposes robot interchangeability as an additional trade-off consideration. Specifically, the work presented here leverages the total interchangeability of robots and develops a series of novel, complete, computationally tractable algorithms to control a team of robots and avoid collisions while retaining a notion of optimality.

This dissertation begins by presenting a robust decentralized formation control algorithm for control of robots operating in tight proximity to one another. Next, a series of complete, computationally tractable multiple robot planning algorithms are presented. These planners preserve optimality, completeness, and computationally tractability by leveraging robot interchangeability. Finally, a polynomial time approximation algorithm is proposed that routes teams of robots to visit a large number of specified locations while bounding the suboptimality of total mission completion time. Each algorithm is verified in simulation and when applicable, on a team of dynamic aerial robots.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Mechanical Engineering & Applied Mechanics

First Advisor
Vijay Kumar

Second Advisor
Nathan Michael

Keywords
Approximation Algorithm, Completeness, Formation Control, Multi-Robot Planning, Task Assignment, Trajectory Generation

Subject Categories
Computer Sciences | Mechanical Engineering | Robotics

SAFE, SCALABLE, AND COMPLETE MOTION PLANNING OF LARGE TEAMS OF INTERCHANGEABLE ROBOTS

Matthew Turpin

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2014

_____
Vijay Kumar, Supervisor of Dissertation
UPS Foundation Professor of Mechanical Engineering and Applied Mechanics

_____
Nathan Michael, Co-Supervisor of Dissertation
Assistant Research Professor of Robotics

_____
Prashant Purohit, Graduate Group Chairperson
Associate Professor of Mechanical Engineering and Applied Mechanics

Dissertation Committee:

Mark Yim, Professor of Mechanical Engineering and Applied Mechanics
Vijay Kumar, Professor of Mechanical Engineering and Applied Mechanics
Nathan Michael, Assistant Research Professor of Robotics
Howie Choset, Professor of Robotics
Camillo J. Taylor, Professor of Computer and Information Science

Acknowledgments

Over the course writing this dissertation, I have had the pleasure of working with many wonderful people on a number of very interesting projects.

I would especially like to thank my advisors Vijay Kumar and Nathan Michael for their support over the course of my Ph.D. They have continually shown interest and enthusiasm in my work and given me a great deal of support.

I thank my committee members, who were extremely helpful in the dissertation writing process and have provided excellent feedback on improving the quality of this work and its presentation.

My NSF graduate research fellowship has given me a tremendous amount of flexibility in choosing research direction. I thank my undergraduate mentors Kevin Lynch and Tom Vose for helping me define and produce my own research at such an early stage in my career that led to my NSF award.

I would like to thank all of my lab mates in the MRSL for providing a fun community where there is always someone willing to offer a hand.

Most of all, I thank my partner Kristen. She is a tremendous inspiration and I could not have made it without her help and support.

ABSTRACT

SAFE, SCALABLE, AND COMPLETE MOTION PLANNING OF LARGE TEAMS OF INTERCHANGEABLE ROBOTS

Matthew Turpin

Vijay Kumar

Nathan Michael

Large teams of mobile robots have an unprecedented potential to assist humans in a number of roles ranging from humanitarian efforts to e-commerce order fulfillment. Utilizing a team of robots provides an inherent parallelism in computation and task completion while providing redundancy to isolated robot failures. Whether a mission requires all robots to stay close to each other in a formation, navigate to a preselected set of goal locations, or to actively try to spread out to gain as much information as possible, the team must be able to successfully navigate the robots to desired locations.

While there is a rich literature on motion planning for teams of robots, the problem is sufficiently challenging that in general all methods trade off one of the following four attributes: completeness, computational scalability, safety, or optimality. This dissertation proposes robot interchangeability as an additional attribute for consideration. Specifically, the work presented here leverages the total interchangeability of robots and develops a series of novel, complete, computationally tractable algorithms to control a team of robots and avoid collisions while retaining a notion of optimality.

This dissertation begins by presenting a robust decentralized formation control algo-

rithm for control of robots operating in tight proximity to one another. Next, a series of complete, computationally tractable multiple robot planning algorithms are presented. These planners exploit the interchangeability property to achieve optimality, completeness, and computational tractability. Finally, a polynomial time approximation algorithm is proposed that routes teams of robots to visit a large number of specified locations while bounding the suboptimality of total mission completion time. Each algorithm is verified in simulation and empirically evaluated on an experimental testbed consisting of a team of dynamic aerial robots.

Contents

Chapter 1: Introduction

Recent years have seen teams of mobile robots introduced in commercial applications ranging from e-commerce order fulfillment [19] to entertainment and advertising [37]. Using more than one robot is appealing since in addition to providing increased computational power, each robot can pass through, explore, or monitor a different area. This allows the team to provide a greater presence than any single robot (or human for that matter) could ever hope to achieve. A team can also perform tasks that individuals cannot perform such as cooperatively transporting payloads. In addition to the added capabilities of a team, multiple robots inherently exhibits robustness to the failure of individual robots since a subset of robots can continue to complete the assigned tasks. These teams of robots may soon save lives operating in dangerous conditions such as automated mining operations or assisting first responders by quickly search through unsafe buildings.

To successfully realize these applications using teams of robots requires robust solutions to a number of challenges spanning many fields of research. These problems include perception, localization, mapping, and planning among others. Perception is the ability of the robots to gather information about the world and reason about this information. Each robot must then use this information to localize itself by identifying its physical position in space. Mapping allows a team of robots to build a model of the physical world. The team of robots needs to be able to plan motions through the world such that robots do not collide with obstacles in the world or with other robots.

While each of these problems merits substantial attention, this dissertation focuses on the problem of planning for the team of robots. This document assumes each robot has a known map of the environment, can accurately localize itself within that map, and reliably communicate with a base station or nearby robots. Under this assumption, planning team actions is the next critical step required.

There are a number of high quality algorithms to plan actions for a single robot in a known environment. However, it is far more challenging to plan actions for a team to navigate safely (without collision) through an environment than it is to plan for each robot individually. In fact, planning time grows exponentially if single agent planning methods are extended to the multi-robot case as a result of the explosion of the size of the configuration space. Due to this exponential growth, other planning algorithms must be used to ensure plans can be computed for more than a couple of robots in a reasonable time.

Depending on the mission objectives, there are many possible methods that can be used to generate these plans. First, consider a team of robots carrying a payload heavier than any one robot can transport. For this objective, it is critical to ensure robots maintain a fixed shape in space with close proximity to other robots. Since local separation is critical, a leader-follower or formation control algorithm could be used. These algorithms have the property that each robot reacts to neighboring robots to ensure local separation is maintained and collisions between robots are avoided. This dissertation will present a decentralized formation control approach that plans trajectories for robots to provide smooth control policies for each robot with guaranteed convergence to the desired shape. This algorithm functions well even on teams of robots with limited computational and

communication capabilities.

Unfortunately, this formation control approach relies on each robot being physically close to its desired location in the formation to ensure collision avoidance. However, if the robots are interchangeable, each robot can be reassigned to a new spot in the formation that is close to where it actually is. By incorporating this interchangeability, a new method is presented in this document to simultaneously assign robots to goals and generate plans to get them safely to these specified goals. This method works for teams of robots operating in open spaces without obstacles and a modification can be made to transform the algorithm into a decentralized algorithm only considering local information.

So far, these approaches assume a simple, unobstructed environment. An alternate approach must be used to navigate a team of robots to a set of goal locations in obstacle-filled settings. While there are settings that require each robot to visit a specified goal location and in particular for robots with different abilities, this dissertation focuses on interchangeable robots. Utilizing interchangeable robots allows an assignment of robots to goals ensures a collision-free plan will be found if one exists.

Finding plans such that the team of robots can be used to search a known environment such as a damaged building requires a slight change to this problem. In this mission, each goal location only needs to be visited momentarily and any robot can visit any goal. It is often the case that the total mission time should be minimized as opposed to total robot distance traveled. Due to the high number of possible assignments and orderings, an approximation algorithm is presented that results in a close-to-optimal solution.

This dissertation contributes to the existing body of research by presenting a distributed formation control algorithm that provides dynamically feasible actions for a team

of robots as well as a series of computationally tractable, complete, planning algorithms that provide guarantees on optimality. In each problem statement except for formation control, these strong guarantees are made possible by exploiting the property of interchangeability within a team of robots. Preserving the completeness property is critical to ensure mission completion whenever possible. Existing planners without completeness guarantees often fail in environments that are of most interest for large numbers of robots such as operating in close proximity and in tight spaces. By ensuring scalable computational complexity as the number of robots increases, applications such as search and rescue will never be at risk of grinding to a halt when deployed in an unknown setting.

Existing solutions to the problems of single- of multi-robot planning and the benefits and drawbacks of each approach are considered in Chapter 2. After this review, Chapter 3 presents the first contribution of this dissertation, a decentralized finite horizon leader follower algorithm that guarantees convergence in trajectory space as opposed to the general state space. The second contribution of this thesis, a complete multi-agent motion planning algorithm with polynomial complexity bounds for robots in an obstacle-free environment, is developed in Chapter 4. Next, an extension in Chapter 5 allows motion planning in obstacle filled environments. Finally, a constant factor bounded suboptimality approximation algorithm for generating plans to visit many more goals than robots is presented in Chapter 6.

Many of the algorithms presented in this paper are validated experimentally using a teams of robots. In particular, flying quadrotor Micro Aerial Vehicles (MAVs) are used. These small, lightweight vehicles are capable of high speed flight over rough terrain that would be impossible for ground robots to pass through. Their ability to hover in place

gives the MAVs the unique versatility to operate indoors where fixed-wing MAVs would not have room to maintain the forward velocity necessary to stay in the air. Appendix A reviews the basic dynamics of quadrotor MAVs.

Chapter 2: Scientific Background

This chapter reviews a number of basic planning concepts that are referenced in later chapters. Additionally, a literature review positions this dissertation with respect to state of the art planning methods for teams of robots. Section 2.1 begins with a review of some basic properties of planning paths and trajectories for a single robot. Then, multi-robot planning and the associated challenges are considered in Sect. 2.2 and formation control of teams is discussed in Sect. 2.3. Next, the concept of utilizing robot interchangeability is introduced in Sect. 2.4. Finally, Sect. 2.5 studies how goals can be assigned to robots and ordered when there are far more goals than robots.

2.1 Single Agent Trajectory Planning Review

Depending on the robot, environment, and assumptions made, there are numerous motion planning algorithms to navigate a single robot through an environment. LaValle [43] presents in-depth analysis of a great number of methods and discusses how to choose an algorithm based on the setting. This work also details each method's implementation and can be an invaluable resource.

This dissertation uses sampling based motion planning, which has two general components. The first component is sampling: constructing a graph $G = (E, V)$ with vertices $v \in V$ representing collision-free states and edges $e \in E$ representing collision-free transitions between vertices. The second component is finding a path through this graph from

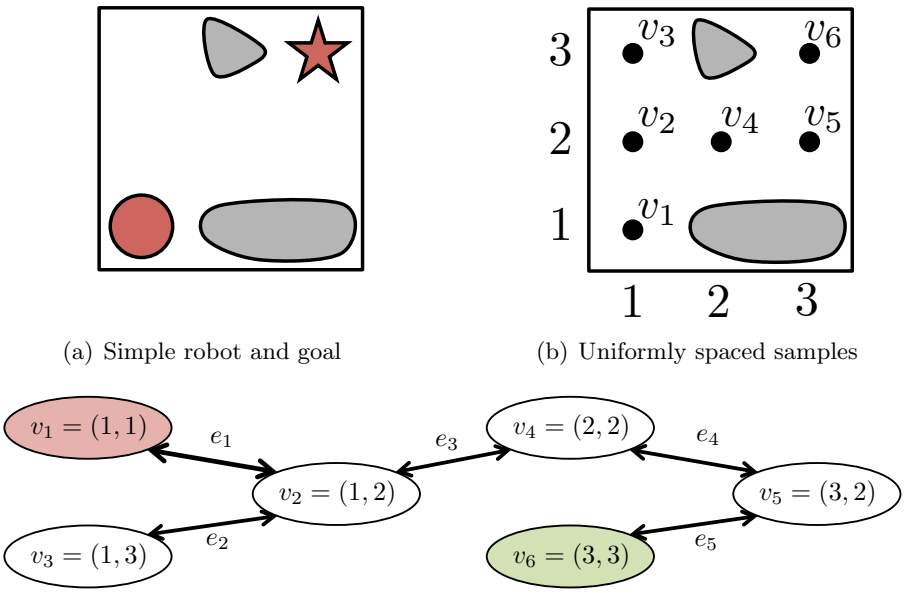
the robot's start to its goal.

There are many methods to create the underlying graph G including grid based methods, Probabilistic RoadMaps (PRM) [32] and visibility graphs [51]. Grid based methods sample the environment at uniform intervals and an edge connects each sample to a fixed number of neighbors that can be reached safely. PRMs are generated by randomly sampling the environment and attempting to connect neighboring samples. By correctly using PRMs, high dimensional spaces can often be searched more efficiently than a uniform grid. Visibility graphs can be used to generate minimum Euclidean distance paths through polygonal two-dimensional environments by connecting obstacle vertices with straight lines to yield the globally optimal solution to the planning problem. This section presents an example using grid based methods, however other sampling methods will be used later in this dissertation.

Consider the example two-dimensional circular robot in Fig. 2.1(a) that lives in a plane. A simple unit-spaced grid can be used to sample the space in Fig. 2.1(b). Any sample that would result in a collision with an obstacle is not considered. Each valid sample is represented by a vertex and for this simple example, there are six vertices in the graph:

$$v_1 = (1,1) \qquad v_2 = (1,2) \qquad v_3 = (1,3) \qquad v_4 = (2,2) \qquad v_5 = (3,2) \qquad v_6 = (3,3)$$

Then, an edge is added between each pair of neighboring vertices if a path can be generated that connects the vertices without collision. In the simple example, there are 5 undirected



(a) Simple robot and goal

(b) Uniformly spaced samples

(c) Graph representation

Figure 2.1: A simple two-dimensional robot in a plane is shown in Fig. 2.1(a) where obstacles are shown in grey and the goal for the robot is denoted with a star. Figure 2.1(b) displays a simple, uniformly spaced grid to sample the space. The graph representing the vertices and corresponding edges is shown in in Fig. 2.1(c), where the red vertex is the start vertex and the green vertex is the goal vertex.

edges (motion in either direction is possible) that can be added between direct neighbors:

$$e_1 = (v_1, v_2) \qquad e_2 = (v_2, v_3) \qquad e_3 = (v_2, v_4) \qquad e_4 = (v_4, v_5) \qquad e_5 = (v_5, v_6)$$

See Fig. 2.1(c) for a visualization of the graph for this simple example. Each edge has an associated *cost* that is related to the desirability of traversing that edge. If the distance is used as the cost function in Fig. 2.1(b), the cost of each edge is one.

Once the graph has been constructed, a graph search algorithm can be utilized to find minimum cost paths through the graph. Dijkstra's algorithm [15] is a well known algorithm for finding the minimum cost path through the graph from a given vertex to all other vertices. A* [27] is a variant of this approach which utilizes a best-first heuristic to often provide better computational performance than Dijkstra's algorithm in practice.

A single agent planner is considered *complete* [43] if it has the following two properties. First, if some solution exists, the planner must successfully return a solution. Second, if a solution does not exist, the planner can recognize this and correctly returns that no solution exists. A planner which is *resolution complete* will find a solution if one exists if the resolution of the underlying graph is sufficiently fine. Both Dijkstra's algorithm and A* are optimal and complete algorithms that can be computed efficiently.

For the graph in Fig. 2.1(c), the plan, or sequence of vertices, that results in the lowest cost plan from the start to goal is $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$. This corresponds to the robot moving from its initial state at $(1,1)$ to $(1,2)$ to $(2,2)$ to $(3,2)$ to the goal at $(3,3)$. In this case, it is the only plan that does not revisit any vertices. The cost of this plan is the sum of the cost of the edges used $\{e_1, e_3, e_4, e_5\}$, or a cost of 4.

This dissertation distinguishes between a path through the graph and a trajectory. A *trajectory* is defined as the continuous mapping from a given time instant to the desired state $\mathbf{x}(t)$. This state may consist of the position, orientation, or velocity of the robot.

The individual robot dynamics will determine how to best compute trajectories between states. The most basic approach for a simple, fully actuated, first order robot is to linearly interpolate between states with constant speed. For example, a trajectory from $v_1 = (1,1)$ to $v_2 = (1,2)$ in the example in Fig. 2.1 could be $\mathbf{x}(t) = 1\hat{\mathbf{i}} + (1+t)\hat{\mathbf{j}}$, $t \in [0,1]$. The distinction of trajectories from paths is necessary because a robot cannot instantaneously transport between vertices. The trajectory generation problem is especially challenging for dynamic robots that cannot always follow straight line trajectories. As a result of using quadrotor MAVs throughout this dissertation, minimum snap trajectory planning [53, 74] will be used since it is well suited to the dynamics of these robots.

2.2 Planning for Multiple Robots

This section outlines two common strategies used to plan collision-free trajectories for a team of robots. The first class of planning algorithms are *decoupled* algorithms that plan for each robot while disregarding other robots. These individual solutions are then merged to form a plan for the team. The second type of algorithms are *coupled* planners that combine the team of robots into a single higher dimensional agent. Each state of the higher dimensional agent can be transformed back to the original robots to generate the plan for the team.

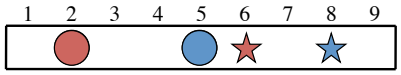
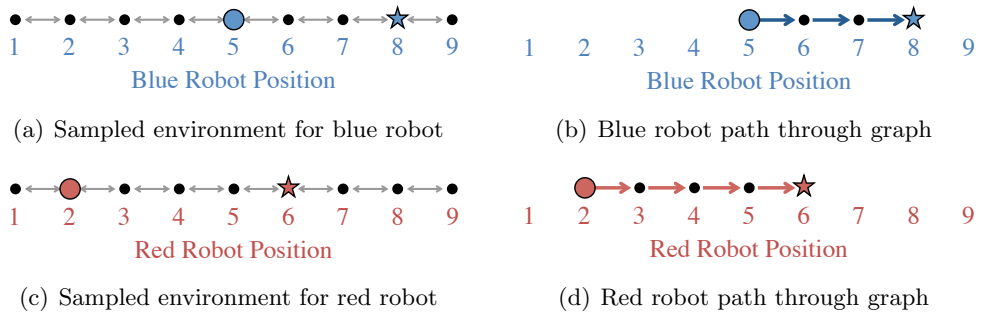


Figure 2.2: Multiple robots with specified goals. Each robot will attempt to navigate to the appropriately colored goal location without colliding.

As an illustrative example, consider the team of two one-dimensional circular robots in Fig. 2.2. In this example, the desired output is a plan for the red robot to navigate to the red goal and the blue robot to the blue goal. A trajectory needs to be designed for each robot such that it reaches the desired goal location without colliding with the other robot. While this may appear to be a simple problem, outlining how coupled and decoupled planners work illuminates the benefits and challenges of each method.

Decoupled methods begin by attempting to find a solution for each robot individually while ignoring all other robots [20]. As discussed in Sect. 2.1, there are many possible methods to plan for each robot in the team. In this section, sampling based method will be used for consistency.

To demonstrate how a decoupled method works on the basic example in Fig. 2.2,



(a) Sampled environment for blue robot

(b) Blue robot path through graph

(c) Sampled environment for red robot

(d) Red robot path through graph

Figure 2.3: Solving for each robot using a decoupled method. In this method, each robot plans as if it were the only robot in the system.

initially consider the blue robot on its own. As noted before, the first step in a sampling based method is to construct a graph consisting of a number of vertices and edges connecting the vertices that represent the space. Unit sampling results in possible states, or vertices, that the blue robot can occupy as $V = \{v_1 = (1), v_2 = (2), \ldots, v_9 = (9)\}$. Each of these states can be reached from its nearest neighbor such that there are 8 edges: $e_1 = (v_1, v_2), e_2 = (v_2, v_3) \ldots e_8 = (v_8, v_9)$. This graph is visualized in Fig. 2.3(a). The next step in the sampling based method is to plan a path through the graph and this is shown in Fig. 2.3(b). For this path, the blue robot will move from $5 \rightarrow 6 \rightarrow 7 \rightarrow 8$. Using the same process results in generating the graph for the red robot in Fig. 2.3(c) and the solution in Fig. 2.3(d).

Now, an attempt to merge the plans is carried out. The first step is to transform the plans into time-parameterized trajectories as outlined in Sect. 2.1. For simplicity, each robot will either remain motionless or move with constant speed of one unit of distance in one unit of time. A typical method for merging plans is to prioritize robot motion such that the robot with highest priority moves first and then the second highest priority robot

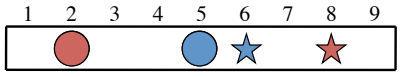


Figure 2.4: The same problem as in Fig. 2.2, but with goals exchanged between robots.

and so on. If the blue robot moves first, it will reach its goal without colliding with the red robot. Then, if the red robot waits until the blue robot is at its goal, the red robot can reach its goal without colliding with the blue robot. Following these trajectories, the locations of each at each integer time is as follows:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
red robot position	2	2	2	2	3	4	5	6
blue robot position	5	6	7	8	8	8	8	8

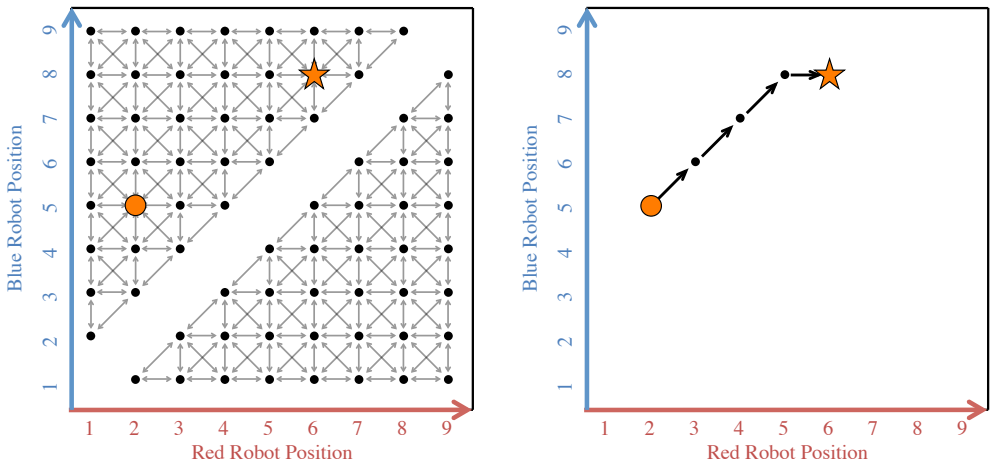
Alternatively, the red robot can begin moving sooner than waiting until the blue robot has finished:

	t_0	t_1	t_2	t_3	t_4
red robot position	2	3	4	5	6
blue robot position	5	6	7	8	8

At each time instant, note that the two robots are not in collision. It should also be clear that in this example, the robots will not collide between time steps.

While decoupled approaches often work quite well when robots do not interact much throughout their trajectories, they are unable to provide completeness guarantees. To demonstrate this lack of completeness, consider the example problem with goals reversed in Fig. 2.4. While each robot can plan a path individually to reach the goal, there is no method that will guarantee both robots can reach their goals without collision. The planner will not be able to return a collision-free set of trajectories, but will not be able to guarantee that no solution exists.

In contrast to a decoupled planner, coupled methods do not plan for each robot on a separate graph. Instead, the states of individual robots are concatenated to form a single



(a) Coupled graph (b) Path through coupled graph

Figure 2.5: The coupled graph for the red and blue robots is presented in Fig. 2.5(a). The starting state of the coupled robot is identified by the orange circle and the coupled goal is denoted by the orange star. The optimal path through the graph is displayed in 2.5(b).

multi-robot state. For the two one-dimensional robots in Fig. 2.2, a coupled planner considers the problem as that of a single two-dimensional robot. The coupled initial state is $(2, 5)$ with the first coordinate representing the red location and the second coordinate representing the blue robot. Similarly, the goal state for the coupled system is $(6, 8)$. Next, a single two-dimensional graph is generated to represent the system. The graph is constructed by sampling a regular grid and then connecting samples in this higher dimensional space. This higher dimensional graph is visualized in Fig. 2.5(a). Note that vertices along the diagonal are removed since these vertices represent the robots occupying the same location and therefore result in a collision between robots.

At this point, it is clear that the graph search methods described in Sect. 2.1 can be used to find a solution from the initial state of this multi-robot state. The minimum cost solution is displayed in Fig. 2.5(b) and has a solution of $(2, 5) \rightarrow (3, 6) \rightarrow (4, 7) \rightarrow$

$(5, 8) \rightarrow (6, 8)$. To execute this plan, each state in the higher dimensional space can be mapped back to the original one-dimensional problem for two robots:

	t_0	t_1	t_2	t_3	t_4
red robot position	2	3	4	5	6
blue robot position	5	6	7	8	8

Since each state in the graph of the coupled agent is collision-free and each edge represents a collision-free trajectory, this plan is guaranteed to be collision-free.

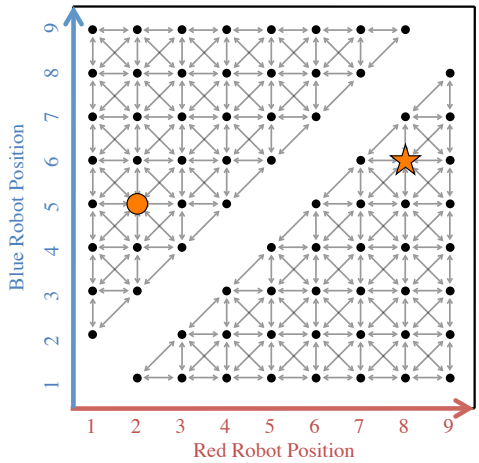


Figure 2.6: The coupled graph for Fig. 2.4. There is no solution to reach the goal state from the initial state.

A benefit of coupled methods over decoupled methods is that coupled methods preserve the completeness guarantees of the single agent planner used. If goal locations are switched as in Fig. 2.4, the coupled planner generates the graph in Fig. 2.6 and is able to correctly identify that there is no solution from the initial configuration to the goal configuration.

Unfortunately, the dimensionality of the graph for coupled planners grows linearly in the number of robots. As a consequence, the number of samples required grows exponentially with the number of robots [20, 43]. As a point of comparison, the coupled graph in Fig. 2.5(a) has 72 vertices and 210 edges after removing collision edges and there are only

9 vertices and 8 edges for each decoupled robot. While probabilistic sampling methods (eg. PRM) or probabilistic planners (such as RRTs) are more successful at in high dimensional spaces than grid based methods, they simply cannot handle the dimensionality of the search space for medium to large teams of robots.

Recently, there has been interest in combining the best elements of coupled and decoupled planners. Optimal decoupling [91] attempts to only couple robots that must be coupled and minimizes the highest coupled state space. Similarly, subdimensional expansion [95, 96] can be employed to mitigate increased computational complexity by selectively expanding the higher-dimensional state space only as required. Each of these methods can have exponential complexity in the number of robots if robots interact substantially.

Other notable multi-robot planing approaches switch robot positions until robots can easily navigate to goal positions without collision [52, 69]. However, many of these rule based methods can generate poor trajectories consisting of all robots initially moving away from their goals before one by one navigating to the correct goal locations. The operations research community also considers planning for multiple vehicles to move through a space while avoiding collision. There is a wide variety of modifications that can be used including sequential routing [34, 35, 57], integer coupled methods [62], and those without optimality guarantees [24, 39, 70]. An alternative strategy is to formulate multi-robot coordination and collision avoidance as reactive control or local coordination problems. The decentralized collision avoidance method proposed in [90] enables a robot team to navigate to an assigned set of goals and scales well with the number of agents but lacks safety and optimality guarantees for systems with higher-order dynamics.

2.3 Formation Control

An alternative to planning unique actions for each robot in the team is to maintain a specified formation and then plan a single basic group motion. Planning in this manner can reduce the computational complexity to the equivalent of planning for a single agent, but requires a robust control approach to maintain the formation. Some examples of tasks that work well for formation control include cooperative manipulation of large payloads either via grippers or cables, reconnaissance where imagery from different sensors needs to be registered and stitched together, and persistent surveillance where robots maintain coverage of a dynamically changing environment.

Depending on the mission tasks, there are various methods for determining the formation specification and planning the formation's motion. The desired formation specification often consists of relative position or bearing information and might be time varying. Virtual Structure (VS) methods are some of the most basic schemes used. In VS approaches, robots are virtually embedded in rigid shape and then single robot trajectory generation methods can be used to plan bulk motion. The group trajectory may specify a plan for the centroid of robots, the position of a single "leader" robot, or the position of a virtual robot. Generating the group trajectory for a rigid formation is equivalent to planning for a single robot (see Sect. 2.1) and can be computed quickly and reliably.

While centralized leader-follower schemes exist, they are limited in a few ways. First, a centralized approach requires communication with every robot in the team. This may not be possible for teams that are spread out over large areas. Instead, a decentralized control approach only requires information from neighboring robots. Similarly, using

the information of neighboring robots allows relative separation between neighbors to be directly controlled. Using local information can result in a smaller relative error, allowing tighter formations without risk of collision.

Similar to results presented in this dissertation, decentralized control of aerial vehicles following the leader-follower framework is discussed in [26], where the performance of the controllers are analyzed through experimentation. In addition to considering stability and convergence properties, it is possible to derive conditions that ensure feasible formation structures based on individual robot capabilities [80]. Input-to-state stability of formations is discussed in [81], by considering the construction of graphs from primitives which provide known stability properties. The authors of [21, 65] consider information flow in feedback-based controllers and the consequential effects on the stability of the system to converge to the desired formation. One example distributed controller for trajectory tracking by a team of robots maintaining a rigid VS is discussed in [17].

In addition to maintaing the formation, some works study transitions between formation specifications. One possible approach to plan between a number of formation descriptions consisting of distance and bearing information is analyzed in [14]. The authors of [63] develop stabilizing controllers for driving a formation of robots to rotate, translate, expand, or contract based upon a VS representation. In [54], the authors consider the controllability of state-dependent dynamic graphs. Consensus problems on formation graph structures with switching topologies and time-delays are discussed in [66].

2.4 Robot Interchangeability

Some applications of teams of robot require each robot in a team to reach a robot specific destination. For example, consider an assembly task where robots are delivering parts to the correct assembly area. Imagine one robot is carrying widget a, which is required at site A, and another robot with widget b, required at site B. Each robot clearly must navigate to the correct location to satisfy the objective. The general problem of matching task to robots is discussed in [25].

However, there are also a large number of uses of multi-robot systems where any robot visiting every site will satisfy all mission objectives. In the previous example, if every robot had widget a and all sites required that widget, it is unimportant which robot ends up at which goal. It is often the case that all robots contain the same set of sensors and the sensors are the "widgets" to position. For example, the use of a team of identical robots to position cameras and microphones for first responders or law enforcement at points of interest is indifferent to which robot is at which location. Other representative examples of such applications include automated storage and product retrieval in warehouse applications [19] and automated structure construction in which robots are commanded to fetch and place parts [48, 97] or when the robots themselves are used as structural elements of a larger construction [64, 67].

In every application where robots are interchangeable, the assignment of robots to goals provides an additional degree of freedom to consider in the design of the algorithm. For completely interchangeable teams, any assignment is sufficient to meet the mission objectives, but clearly some assignments are better than others. Because system demands

depend on the particular event and can change quickly, it is important to not only quickly choose or compute an assignment, but also choose a high quality assignment that will quickly achieve the mission goals.

One of the most commonly used assignment problem statements is that of the linear assignment problem. This problem is stated as follows: given N workers and N tasks where each worker has a cost of achieving each task, find the minimum cost assignment such that each worker is assigned one task and each task is completed by one worker. Optimal solutions to the linear assignment problem have been developed in the operations research community [41].

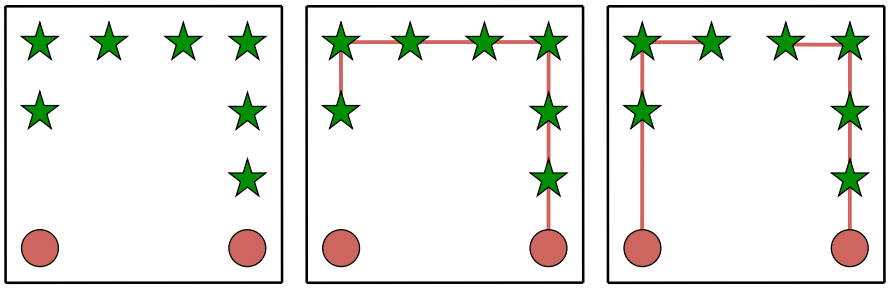
Some researchers have explored using the assignment which minimizes the sum of Euclidean distance traveled for a set of completely interchangeable goals [31, 77]. Ignoring collisions, this objective function will have the least possible robot motion for optimal system efficiency. Similarly, [49] presents a hybrid method for the Euclidean assignment algorithm for very large numbers of robots with both a global and local approach. [2] finds a suboptimal assignment which seeks to minimize distance squared. Each of these methods does not provide either completeness or collision avoidance guarantees.

An interesting approach is to consider coupling assignment with planning collision-free trajectories for the team of robots to reach those objectives. A suboptimal approach to solve this problem is presented in [36], which proposes a centralized algorithm to create collision-free paths and solve the assignment problem for 2-dimensional robots. Other recent work incorporating multi-robot task assignment into trajectory planning includes [36, 49, 50]. As shown in [98], using network flow algorithms to design collision-free trajectories can yield substantially improved performance for kinematic agents.

2.5 Vehicle Routing

The problem of allocating a team of robots to visit a large set of spatially distributed tasks without any preference for which robot visits which goal or in what order is common in both the operations research and multi-robot community. In the case that a single robot is required to visit more than a single goal, not only does the route to get to each goal need to be planned, but the order in which to visit goals must be considered. This problem is actually quite similar to the well-known Traveling Salesman Problem (TSP). The TSP seeks to visit a number of cities and return to the original location while traveling the least total distance. To formalize this idea, define a Hamiltonian cycle as a loop of vertices in a graph such that every vertex is visited exactly once. The TSP is defined to find the lowest cost, or minimum weight, Hamiltonian cycle in a graph. A distinction between a Hamiltonian path and a Hamiltonian cycle in that the path visits each location but does not form a loop. Unfortunately, finding either the minimum cost Hamiltonian cycle or Hamiltonian path is NP-Hard.

Operations research literature contains a number of exact and approximation algorithms to plan solutions to the Vehicle Routing Problem (VRP) [18]. These methods generate routes for a team of agents leaving one or more depots, visiting a number of goal locations, and returning back to the original depots. The VRP is especially well suited to shipping industries where trucks are physically leaving depots at the start of a shift and returning to the same depot at the end. The substantial body of literature on the VRP contains a huge variety of differences in problem formulation and assumptions that lead to drastically different solutions. These variations include the Pick-up and Delivery



(a) Routing problem (b) Min cost solution (c) Min-max cost solution

Figure 2.7: Shown in 2.7(a) is a set of two robots and seven goals that need to be visited be either robot. Figure 2.7(b) shows the typical minimum cost solution that requires one robot to travel 7 spaces. Figure 2.7(c) displays the minimum maximum solution. The min-max solution has a total cost of 8, but will complete after each robot travels 4 units. The minimum cost solution takes nearly twice as long to complete as the min-max cost solution.

Vehicle Routing Problem (PDVRP), the Vehicle Routing Problem with Time Windows (VRPTW), and many others [18].

Typically, VRP solutions seek to minimize the total distance traveled by all robots. In these instances, it is possible that most robots travel a relatively short distance and a few agents must travel much further to ensure all locations are visited. This cost function is unacceptable for time critical applications. Consider the simple example in Fig. 2.7 to demonstrate that minimizing the maximum cost for any robot reduces completion time. The minimize maximum distance formulation of the VRP is referred to as the Minimum Maximum Vehicle Routing Problem (MMVRP) [72].

In the robotics community where robots are not necessarily beginning at a common starting location, the Multiple Depot Vehicle Routing Problem (MDVRP) [60] is a better fit. In the MDVRP, agents are not required to start and finish at one central location, but can instead operate out of up to N unique locations with each agent returning to its original depot. Combining these two problems yields the Minimum Maximum Multiple

Depot Vehicle Routing Problem (MMMDVRP) [9], which is especially useful in robotics.

Another related problem is the orienteering problem that is especially useful for battery constrained robots. The orienteering problem is defined as follows: given time T, navigate to as many of M specified points as possible [92]. The Team Orienteering Problem (TOP) uses N agents to visit as many of the waypoints as possible in the allotted time. If the time is sufficient to visit all of the waypoints, the solution very closely resembles the VRP solution. However, the VRP, and TOP literature focuses on assignments which typically ignore the collision avoidance constraint, which must be considered when routing teams of robots.

Chapter 3: Formation Control

As discussed in Sect. 2.3, achieving and maintaining a desired formation of autonomous robots has wide appeal in increasing system redundancy, efficiency, and flexibility. By ensuring a formation is maintained, the problems of controlling a team of robots becomes the more computationally manageable task of maintaining a single large rigid body. However, for dynamic robots such as quadrotor MAVs, a closed loop control policy must be used to ensure communication errors or execution noise does not drive the system unstable. This chapter presents a decentralized formation control algorithm that guarantees convergence to the globally optimal solution for sufficiently fast communication rates.

Formation control approaches can typically be classified into one or more of the following categories: leader-follower [13], virtual-structure [47], and behavior-based control [6]. This chapter considers virtual-structure formations, but allows the structure, or shape, to be time varying. In addition to the variety of control schemes, there are a great number of sensing and information gathering modalities used to maintain the formation. Commonly used sensors include range only ultra wideband radios [16], bearing only cameras [22], wireless communications [1], or a combination of these sensing modalities.

Many approaches rely on a centralized robot manager to plan motions for each agent of the team. In many cases, it is desirable for a team of robots to safely maintain the formation using only local data. Imagine a school of fish or migrating geese maintaining

the shape of the flock by rely on the information gleaned from their neighbors. By utilizing only information available to each robot allows the system greater flexibility and scalability since no centralized solution is required [55, 59, 99].

In these networked systems, data latencies and dropped packets over wireless links are hard to predict and can have a dramatic negative effect on the overall system performance. This is especially true when delayed data is substantially different than expected. Data discontinuity can even cause robots to become unstable or crash. Despite failure mitigation through intelligent planning and reactiveness, there are still a myriad of ways a robot may separate from the team. These include robot collisions with obstacles, people, or other robots. There are also unexpected hardware failures of robots such as battery deterioration or mechanical breakage that removes a robots utility to the team. An intelligent local control policy must be designed to handle these network and robot failures to ensure the success of a decentralized policy.

This chapter proposes a decentralized approach for controlling a team of robots to maintain a tight formation while moving quickly and precisely through the environment. In addition to utilizing a time-varying virtual-structure, this proposed method makes use of a leader robot to plan the formation bulk motion. While this chapter focuses on a team of quadrotor MAVs, the approach presented is general enough to be applied to any team of differentially flat robots. As a result of the complex dynamics of the quadrotor robots outlined in Appendix A and the associated short time scales, reliance on typical causal feedback policies does not yield acceptable performance. Planning reactively in state space is simply insufficient to maintain high quality formation control for cooperative payload transport or maintaining fixed imaging distances. Instead, the robots must be

planned for in trajectory space using model predictive, or receding horizon control [30] to plan trajectories for each robot over a short time horizon. More specifically, decentralized model predictive control [33, 76] is used since the robots are controlling using only local information.

It is assumed that every robot has knowledge of its place in the formation and a single bulk trajectory is designed by the leader robot. Using the formulation controller presented in this chapter, each robot communicates with and updates its neighbors with its own motion plan. Then each robot plans its own local optimal trajectory based on state information about its neighbors and the last communicated plan of the neighbors. This iterative sequence of communication and re-planning is continued until mission completion.

A two-component representation for the formation consisting of the ensemble shape and a dependence measure for each communication link is presented in Sect. 3.1. Section 3.2 details control laws for formation control with a team of aerial robots and discusses considerations for improving controller performance given degrading network performance and robot failures. Section 3.3 discusses the expected convergence properties of the proposed controllers. The chapter concludes with an series of simulations in Sect. 3.4 and an experimental study in Sect. 3.5 to evaluates performance under varying system conditions. In both simulation trials as well as experimental evaluation of this approach on a team of quadrotors, suitable performance is maintained as the formation motions become increasingly aggressive and as communication degrades or with individual robot failure.

The research contained in this chapter was initially published in [82–84].

3.1 Modeling the Robot Formation

The virtual-structure formation control approach used in this chapter implies that each robot in the team acts as a particle embedded in an imaginary solid object. As such, a formation specification can be defined by a set of relative position vectors between robots. In this chapter, the imaginary solid object is deformable and therefore a time-varying *shape vector* $\mathbf{s}_{ij}(t) \in \mathbb{R}^3$ is defined so the there is no ambiguity in formation shape. This is represented as the vector from the location of robot i to the location of robot j in an inertial frame:

$$\mathbf{s}_{ij}(t) = \mathbf{x}_j(t) - \mathbf{x}_i(t) \tag{3.1.1}$$

Figure 3.1 displays a number of shape vectors. In a team of N robots, there are N^2 shape vectors that can be defined with $\binom{N}{2}$ pairwise distances. However, many of these shape vectors are coupled and must satisfy the following properties to produce a consistent formation:

$$\begin{aligned} \mathbf{s}_{ik}(t) &= \mathbf{s}_{ij}(t) + \mathbf{s}_{jk}(t) \\ \mathbf{s}_{ij}(t) &= -\mathbf{s}_{ji}(t) \end{aligned} \tag{3.1.2}$$

In fact, once $N-1$ unique shape vectors have been specified, the remaining shape vectors that satisfy the properties in (3.1.2) can be directly computed. In this chapter, define the position of each robot in the formation relative to a single reference location. This is not a limiting property and ensures all agents always have the information required to compute relative shape vector with any communicating neighbor. For consistency, use robot 1 as the reference in the formation. Even if a set of robots are far from and have never communicated with the reference robot, they can directly compute any required

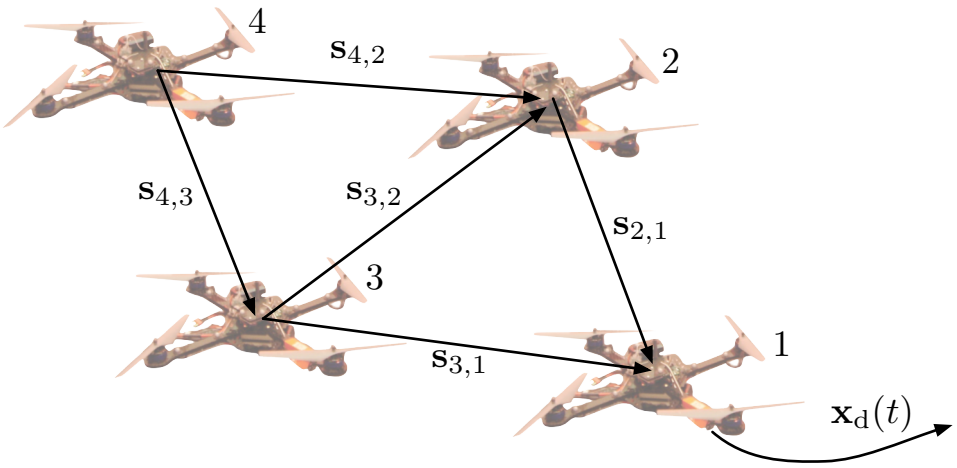


Figure 3.1: The team of robots maintains the formation described by shape vectors $\mathbf{s}_{i,j}$ while the lead robot tracks a desired trajectory $\mathbf{x}_{1(t)}$.

shape vectors using the properties in (3.1.2). It is assumed that shape vectors are chosen to respect pairwise separation constraints and avoid robot-robot collisions.

A single robot is defined as the leader of the team and is responsible for determining the bulk motion of the team. The leader can also be a virtual robot with a reference trajectory for the formation, thus eliminating a single point of failure. This single trajectory can be reliably computed using the methods presented in Sect. 2.1. In particular, the trajectories used for all simulations and experiments in this chapter are computed using the minimum snap methods presented in [53].

The set of all robots with a reliable communication link with robot i is defined as communication set $\mathcal{C}_i$ and can contain up to $(N-1)$ robots. Unlike most leader-follower controllers using hierarchical structures (for example [14]), in this chapter each follower robot can use information received from every other robot in its communication set. Thus follower robot i has the ability to utilize up to $(N-1)$ sets of information.

Since some of robots are further away than other robots, pairwise communication

quality and the importance of maintaing precise relative positioning may vary substantially. As a result, robot i uses a convex combination of dependence weights d_{ij} to reflect the degree to which information from robot j is used:

$$\sum_{j \in \mathcal{C}_i} d_{ij} = 1 \quad \text{and} \quad d_{ij} \geq 0 \tag{3.1.3}$$

These components are then used to construct the $N \times N$ *dependence matrix*, $\mathbf{D}$, whose entries are non-negative and the row sums are all 1. In other words, $\mathbf{D}$ is a right stochastic matrix [42], a fact which will be leveraged later in this chapter. Also note that each robot does not need to know the state of all other robots or even how many robots are in the system, but only informations about its neighbors.

The lead robot has sole dependence on itself and no other robot has any dependence on itself. For example, if robot 1 is the leader, its dependence coefficients are $d_{11} = 1$ and $d_{1j} = 0, \ \forall j \neq 1$. Since every other robot has no dependence on itself, $d_{ii} = 0, \ \forall i \neq 1$. At least one robot must have positive dependence on the lead robot, or $d_{i1} > 0$ for some i. It is assumed that the dependence matrix represents a connected graph. Due to these requirements, it is clear that the dependence matrix $\mathbf{D}$ is not symmetric.

With the exception of the leader robot, define the desired state of robot i as the dependence weighted sum of relative separation:

$$\mathbf{x}_i^{\text{des}} = \sum_{j \in \mathcal{C}_i} d_{ij} \left(\mathbf{x}_j(t) + \mathbf{s}_{ji}(t) \right) \tag{3.1.4}$$

Similar schemes appear in the consensus literature where each robot's desired heading, velocity or acceleration is obtained by a weighted average of information acquired from neighbors [29, 94]. However, in this case the consensus-like rule (3.1.4) is used to define entire trajectories that each robot computes as explained in the next section.

3.2 Formation Control

As previously noted, the lead robot individually computes its desired trajectory and in practice, a quadrotor robot will follow this trajectory using the methods described in Appendix A. For the remaining robots, the error between the current system state and the desired state as defined by $\mathbf{s}_{ij}(t)$ for all i, j is:

$$\mathbf{e}_i(t) = \mathbf{x}_i(t) - \mathbf{x}_i^{\text{des}} = \sum_{j \in \mathcal{C}_i} d_{ij}(\mathbf{x}_i(t) - \mathbf{x}_j(t) - \mathbf{s}_{ij}(t)) \tag{3.2.1}$$

A receding horizon control approach is now proposed to address network delays and other sources of information latency. It is assumed that all robots operate with a synchronized system clock.

Denote t_c as the current time and t_h as the time horizon used for robot trajectory generation. At current time t_c, constraints must be imposed to ensure continuity of the trajectory. For a fourth order system like the quadrotor, this requires continuity up to the third derivative of position. Additional constraints may be added to ensure feasibility of the planned trajectory to respect the actuator limits of the robot and to avoid collision with convex bounding constraints. Constraints to avoid collisions with any obstacle can be included, however these objects can introduce non-convex constraints that are challenging to solve in a computationally tractable manner. Each robot can generate its own trajectory by minimizing the integral of the error squared from (3.2.1) over the interval $[t_c, t_c + t_h]$:

$$
\begin{aligned}
\underset{\mathbf{x}_i(t)}{\text{minimize}} \quad & \int_{t_c}^{t_c+t_h} \mathbf{e}_i(t)^{\mathrm{T}} \mathbf{e}_i(t) dt \\
\text{subject to} \quad & \mathbf{x}_i(t_c) = \text{robot } i \text{ current state} \\
& \dot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current velocity} \\
& \ddot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current acceleration} \\
& \dddot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current jerk} \\
& \mathbf{x}_i(t) \text{ within environment} \\
& \mathbf{x}_i(t) \text{ respects actuator limits}
\end{aligned}
\tag{3.2.2}
$$

This will drive the system to zero error as quickly as physically possible. In the case of the quadrotor, it is beneficial to slow down these convergence dynamics to ensure system smoothness. To reduce the convergence rate, the error derivative terms can be included in the optimization formulation:

$$
\begin{aligned}
\underset{\mathbf{x}_i(t)}{\text{minimize}} \quad & \int_{t_c}^{t_c+t_h} \left[\sum_{k=0}^{3} \kappa_k \, \mathbf{e}_i^{(k)}(t) \right]^2 dt \\
\text{subject to} \quad & \mathbf{x}_i(t_c) = \text{robot } i \text{ current state} \\
& \dot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current velocity} \\
& \ddot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current acceleration} \\
& \dddot{\mathbf{x}}_i(t_c) = \text{robot } i \text{ current jerk} \\
& \mathbf{x}_i(t) \text{ within environment} \\
& \mathbf{x}_i(t) \text{ respects actuator limits}
\end{aligned}
\tag{3.2.3}
$$

where κ_k is the weight applied to the k^{th} derivative and κ_1 is greater than zero to ensure error convergence to zero. Notationally, $\mathbf{e}_i^{(k)}$ is the k^{th} derivative of the corresponding component of the error.

Similar to the trajectory generation for a single robot, the trajectory is defined over the finite horizon, $t_c \leq t \leq (t_c + t_h)$ to be a polynomial of order p :

$$
\mathbf{x}_i(t) = \sum_{k=0}^{p} \alpha_i^k t^k
$$

where $\alpha_i^k \in \mathbb{R}^3$ is the i^{th} robot's k^{th} polynomial coefficients. Similarly define all of the

shape vectors over the interval of the lead robots trajectory $t_0 < t < t_f$ as:

$$\mathbf{s}_{ij}(t) = \sum_{k=0}^{p} \sigma_{ij}^{k} t^{k}$$

where $\sigma_{ij}^{k} \in \mathbb{R}^3$ is the k^{th} polynomial coefficients of the shape between robots i and j.

Then, these polynomial representations can be substituted into the error of robot i in (3.2.1):

$$\mathbf{e}_i(t) = \sum_{j \in \mathcal{N}_i} d_{ij} \left(\sum_{k=0}^{p} \left(\alpha_i^k - \alpha_j^k - \sigma_{ij}^k \right) t^k \right)$$

With these definitions, (3.2.3) can be solved for robot i as a quadratic program in real time to generate a reference trajectory which minimizes the weighted sum of the error and the derivatives of the error.

Thus the basic algorithm for formation control (see Fig. 3.2) is as follows:

1. Each follower robot initializes with a stationary trajectory.

2. The lead robot computes its desired trajectory using the methods outlined in Appendix A and begins to follow this trajectory.

3. Every robot broadcasts its trajectory described by its polynomial coefficients (α_i for robot i) and the appropriate time interval to neighboring robots.

4. Each follower robot recomputes its trajectory for the finite time horizon, t_h, based on the desired shape vectors $\mathbf{s}_{ij}(t)$ and neighboring robots' trajectory information using (3.2.3).

5. Steps 3-4 repeat until either the lead robot issues a termination command which propagates through the system or a pre-specified time limit expires.

A crucial benefit of using this method is that a centralized planner only needs to plan the motion of the group at an abstract level in a low dimensional space. It computes the trajectory for the group (or for a leader of the group), $x_1^{\text{des}}(t)$, and the desired shape $\mathbf{s}_{ij}(t)$ for the formation. Individual robots are only required to compute local trajectories and control inputs.

There are some variations to the basic theme in Fig. 3.2. First, the communications of coefficients α_i are shown for all connections, but there is the possibility that some agents do not communicate with all others. In a real system, there are up to $\binom{N}{2}$ possible communications for a fully connected system every δ seconds. Second, when communication performance degrades (as discussed in the experiments, Sect. 3.5), the robots extrapolate based on the last trajectory specification from each neighboring robot and choose controls accordingly. Finally, when critical failures occur, the robots modify their local dependence measures, d_{ij}, accordingly. Robot failure is detected through the robot sending information that is no longer useful to the team or absence of communication. It is assumed that a robot can identify that it has failed, or it could continue sending incorrect information. This local trajectory control law allows for asynchronous communication and planning and is robust to changing network topology, network latency and dropped messages, as well as critical robot failures.

3.3 Consensus on Trajectories

This section presents the convergence of the proposed scheme and simulation results. The theoretical analysis requires three simplifying assumptions:

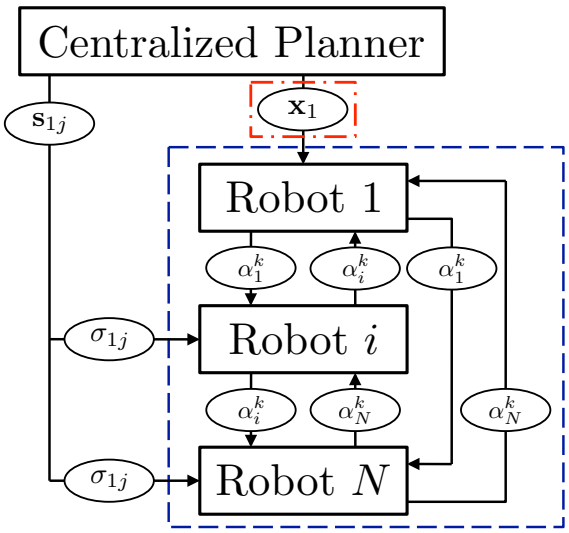


Figure 3.2: System block diagram which clearly shows the centralized planner only needs to initially compute a desired shape $\mathbf{s}_{ij}(t)$ and a group motion $x_1^{\text{des}}(t)$. Chapter A outlines the computation of $x_1^{\text{des}}(t)$, which is shown the red dot-dash box. Section 3.1 formulates the decentralized interactions in the blue dotted box.

(A1) There are no inequality constraints when computing trajectory coefficients α. In such a setting, the trajectories satisfying (3.2.2) are decoupled polynomial functions of time. Hence, the solution for each component of $\mathbf{x}_i(t)$ is independent of other components allowing a decoupled solution to (3.2.2) into three one-dimensional subproblems.

(A2) All robots are homogeneous and use exactly the same optimization criteria, and therefore employ the same optimization coefficients κ in (3.2.2).

(A3) Robot clocks are synchronized and the solution to (3.2.2) can be calculated very fast. If this calculation is instantaneous, all robots can be modeled as having synchronous network updates with all neighbors every δ s.

Some of these assumptions will be relaxed in the experimental studies.

Using (A1) above, the minimization for each component of $\mathbf{x}_i(t) \in \mathbb{R}^3$ for robot i is as follows:

$$\begin{aligned} \underset{\mathbf{x}_i(t)}{\text{minimize}} \quad & \int_{t_c}^{t_c+t_h} \left[\sum_{k=0}^{3} \kappa_k e_i^{(k)}(t) \right]^2 dt \\ \text{subject to} \quad & \mathbf{x}_i(t_c) = f_i^0 \\ & \dot{\mathbf{x}}_i(t_c) = f_i^1 \\ & \ddot{\mathbf{x}}_i(t_c) = f_i^2 \\ & \dddot{\mathbf{x}}_i(t_c) = f_i^3 \end{aligned} \tag{3.3.1}$$

where f_i^k is the current k^{th} time derivative of the component of state of $\mathbf{x}_i(t)$.

Define the coefficient vectors $\boldsymbol{\alpha}_i = [\alpha_i^0, \alpha_i^1, \ldots, \alpha_i^p]^{\mathrm{T}}$, $\boldsymbol{\sigma}_{ij} = [\sigma_{ij}^0, \sigma_{ij}^1, \ldots, \sigma_{ij}^p]^{\mathrm{T}}$, and $\mathbf{t} = [t^0, t^1, \ldots, t^p]^{\mathrm{T}}$. Additionally define $\mathbf{f}_i = [f_i^0, f_i^1, f_i^2, f_i^3]^{\mathrm{T}}$ such that the equality continuity constraints can then be vectorized as $\mathbf{E}_i \boldsymbol{\alpha}_i = \mathbf{f}_i$. The weighted shape variable for robot i is computed as:

$$\boldsymbol{\tau}_i = \sum_{j \in \mathcal{N}_i} (d_{ij}(\boldsymbol{\alpha}_j + \boldsymbol{\sigma}_{ij}))$$

The integrand in 3.3.1 can evaluated and then reformulated to:

$$(\boldsymbol{\alpha}_i - \boldsymbol{\tau}_i)^{\mathrm{T}} \mathbf{H}_i (\boldsymbol{\alpha}_i - \boldsymbol{\tau}_i)$$

where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\tau}_i$ have dimension $(p+1) \times 1$, $\mathbf{H}_i$ is $(p+1) \times (p+1)$. Then the following Quadratic Program (QP) is equivalent to the optimization in 3.3.1:

$$\begin{aligned} \underset{\boldsymbol{\alpha}_i}{\text{minimize}} \quad & \frac{1}{2} \boldsymbol{\alpha}_i \mathbf{H}_i \boldsymbol{\alpha}_i - {\boldsymbol{\tau}_i}^{\mathrm{T}} \mathbf{H}_i \boldsymbol{\alpha}_i \\ \text{subject to} \quad & \mathbf{E}_i \boldsymbol{\alpha}_i = \mathbf{f}_i \end{aligned}$$

where $\mathbf{E}_i$ is $4 \times (p+1)$. This QP can be rewritten as an unconstrained optimization using a vector of Lagrange multipliers, $\boldsymbol{\lambda}_i$:

$$\underset{\boldsymbol{\alpha}_i, \boldsymbol{\lambda}_i}{\text{minimize}} \quad \frac{1}{2} \boldsymbol{\alpha}_i \mathbf{H}_i \boldsymbol{\alpha}_i - {\boldsymbol{\tau}_i}^{\mathrm{T}} \mathbf{H}_i \boldsymbol{\alpha}_i + \boldsymbol{\lambda}_i^{\mathrm{T}} (\mathbf{E}_i \boldsymbol{\alpha}_i - \mathbf{f}_i)$$

The solution to the reformulated optimization problem is given by the solution to:

$$\begin{bmatrix} \mathbf{H}_i & {\mathbf{E}_i}^{\mathrm{T}} \\ \mathbf{E}_i & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{\lambda}_i \end{bmatrix} = \begin{bmatrix} \mathbf{H}_i \boldsymbol{\tau}_i \\ \mathbf{f}_i \end{bmatrix}$$

Using the Schur compliment, the closed form solution for $\boldsymbol{\alpha}_i$ is

$$\boldsymbol{\alpha}_i = (\mathbf{I} - \mathbf{Z}_i\mathbf{E}_i)\,\boldsymbol{\tau}_i + \mathbf{Z}_i\mathbf{f}_i$$

where

$$\mathbf{Z}_i = \mathbf{H}_i^{-1}{\mathbf{E}_i}^{\mathrm{T}}\left(\mathbf{E}_i\mathbf{H}_i^{-1}{\mathbf{E}_i}^{\mathrm{T}}\right)^{-1}$$

Now let $\mathbf{A}$ be a $N(p+1)\times 1$ vector of $\boldsymbol{\alpha}_j$, $\mathbf{G}$ a $N(p+1)\times 1$ vector of $\boldsymbol{\sigma}_{1j}$, and $\mathbf{F}$ a $4N\times 1$ vector of $\mathbf{f}_i$ as follows:

$$\mathbf{A} = [{\boldsymbol{\alpha}_1}^{\mathrm{T}}, {\boldsymbol{\alpha}_2}^{\mathrm{T}}, \ldots, {\boldsymbol{\alpha}_N}^{\mathrm{T}}]^{\mathrm{T}}$$

$$\mathbf{G} = [{\boldsymbol{\sigma}_{11}}^{\mathrm{T}}, {\boldsymbol{\sigma}_{12}}^{\mathrm{T}}, \ldots, {\boldsymbol{\sigma}_{1N}}^{\mathrm{T}}]^{\mathrm{T}}$$

$$\mathbf{F} = [{\mathbf{f}_1}^{\mathrm{T}}, {\mathbf{f}_2}^{\mathrm{T}}, \ldots, {\mathbf{f}_N}^{\mathrm{T}}]^{\mathrm{T}}$$

From (A2), all robots have the same κ weights, which yields $\mathbf{H}_i = \mathbf{H}$, $\mathbf{E}_i = \mathbf{E}$, and $\mathbf{Z}_i = \mathbf{Z}$.

Under assumption (A3), each robot will have to recalculate its controls every δ s. The desired shape trajectory, $\mathbf{s}_{ij}(t)$, is now reparameterized in time to get:

$$\mathbf{s}_{ij}(t+\delta) = {\bar{\boldsymbol{\sigma}}_{ij}}^{\mathrm{T}}\bar{\mathbf{t}}$$

where $\bar{t} = t + \delta$ and $\bar{\mathbf{t}} = [(t+\delta)^0, (t+\delta)^1, \ldots, (t+\delta)^p]^{\mathrm{T}}$. Define the $(p+1)\times(p+1)$ matrix $\mathbf{Y}$ such that the new coefficients for this δ-shifted trajectory are:

$$\bar{\boldsymbol{\sigma}}_{ij} = \mathbf{Y}\boldsymbol{\sigma}_{ij}$$

and correspondingly:

$$\mathbf{T} = \mathbf{I} \otimes \mathbf{Y}$$

In particular, the desired shape at time $t_c + \delta$ is given by

$$\mathbf{G}(t_c + \delta) = \mathbf{T}\mathbf{G}(t_c)$$

According to (A3), all robots follow their trajectory based on the coefficients $\boldsymbol{\alpha}_i$ for δ s. The robots then exchange information with their neighbors and calculate their new coefficients independently. The new coefficients for all robots will be given by the equation:

$$\mathbf{A}(k+1) = \mathbf{M}\mathbf{A}(k) + \mathbf{L}\mathbf{T}\mathbf{G}(k)$$

where $\mathbf{A}(k)$ and $\mathbf{G}(k)$ are the coefficients at time $k\delta$ and

$$\begin{aligned} \mathbf{M} &= (\mathbf{D} \otimes (\mathbf{I} - \mathbf{Z}\mathbf{E}) + \mathbf{I} \otimes \mathbf{Z}\mathbf{E})\,\mathbf{T} \\ \mathbf{L} &= (\mathbf{D} - \mathbf{I}) \otimes (\mathbf{I} - \mathbf{Z}\mathbf{E}) \end{aligned}$$

From this, it can be shown that:

$$\mathbf{A}(k) = \mathbf{M}^k\mathbf{A}(0) + \sum_{j=0}^{k-1} \mathbf{M}^j\mathbf{L}\mathbf{T}^{(k-j)}\mathbf{G}(0) \tag{3.3.2}$$

Theorem 3.3.1 proves that the trajectories of the individual robots, given by the coefficients in $\mathbf{A}$ converge to the centralized optimal $\mathbf{A}_c$ in the limit as communications delay δ approaches zero.

Theorem 3.3.1. *Under Assumptions (A1-A3), as the time between updates, δ, tends to zero,* $\lim_{k\to\infty} \mathbf{A}(k) \to \mathbf{A}_c$.

Proof. As δ approaches zero, the reparametrization in shape can be ignored such that:

$$\lim_{\delta\to 0} \mathbf{Y} = \mathbf{I}$$

which in turn means $\mathbf{T} = \mathbf{I} \otimes \mathbf{I}$. Taking these limits, (3.3.2) can be converted to:

$$\mathbf{A}(k) = \mathbf{Q}^k\mathbf{A}(0) + \sum_{j=0}^{k-1} \mathbf{Q}^j\mathbf{R}$$

where

$$\mathbf{Q} = \mathbf{D} \otimes (\mathbf{I} - \mathbf{ZE})$$

and

$$\mathbf{R} = ((\mathbf{D} - \mathbf{I}) \otimes (\mathbf{I} - \mathbf{ZE}))\mathbf{G} + (\mathbf{I} \otimes \mathbf{ZE})\mathbf{A}(0)$$

Note that for $k > 0$:

$$\mathbf{Q}^k = \mathbf{D}^k \otimes (\mathbf{I} - \mathbf{ZE}).$$

Because $\mathbf{D}$ is a right stochastic matrix, from the Perron-Frobenius theorem the largest absolute value of an eigenvalue is always 1 [42]. Additionally, this Perron-Frobenius eigenvalue corresponds to the eigenvector $\mathbf{1}$. All other eigenvalues are guaranteed to be less than 1 in magnitude and

$$\lim_{k\to\infty} \mathbf{D}^k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

Thus $\mathbf{Q}^k$ converges to a constant and this in turn implies $\lim_{k\to\infty} \mathbf{A}(k) \to \mathbf{A}_c$, a constant. In other words, the robots reach consensus on their trajectories. □

3.4 Simulations Results

A series of simulations following the trajectory generation method outlined in Sect. 3.2 are presented in this section. Each simulation makes use of all of the assumptions (A1) - (A3). To closely model what actual quadrotors are able to achieve, the following values are used: $p = 10$, $\kappa_0 = 1$, $\kappa_1 = 1$, $\kappa_2 = 0.25$, $\kappa_3 = 0$, $t_h = 2$ sec, and $t_f = 8$ sec. Initial conditions $\mathbf{f}_i$ are randomly chosen values with zero mean and standard deviation 1 with appropriate SI units.

The first simulation in Fig. 3.3 verifies that this method produces high quality convergence to the desired shape for a team of four fourth order robots. This simulation uses a fully connected and equally weighted dependence matrix:

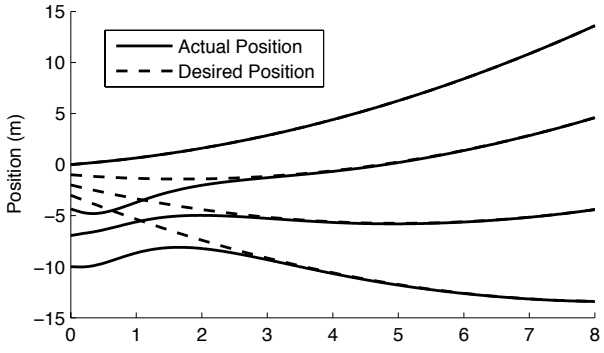
$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

and communications time $\delta = 0.1$ sec.In order to show a formation which changes its shape over time, $\boldsymbol{\sigma}_{1i}$ is selected for all agents such that the desired formation is expanding. Additionally, the leader's trajectory is chosen to be accelerating in the positive direction. Initial randomly chosen positions f_i^0 are shifted to avoid collision, although a more formulaic collision avoidance method is considered in Chapter 4.

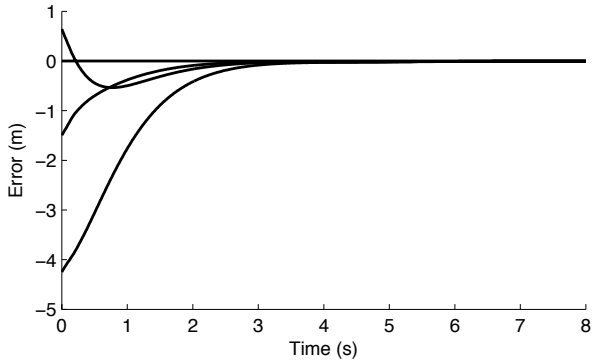
The second simulation in Fig. 3.4 studies the effect of changing δ on the convergence properties of the system. The lead robot's trajectory, $x_1^{\text{des}}(t)$, is held constant at zero and the shape vectors are chosen to be a constant unit distance from each other, $\mathbf{s}_{ij}(t) = 1 \ \forall j = i + 1$. Initial randomly chosen positions f_i^0 are shifted to avoid collision. $\mathbf{D}$ is designed such that the system is a chain so that robot i only broadcasts to robot $i + 1$, a classical leader-follower formation. This results in a dependence matrix for 4 robots of:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

This simulation is carried out using $\delta = \{0.45, 0.15, 0.05\}$ sec. As in the previous simulation, each agent's error converges to zero. Clearly and intuitively, trajectories converge more rapidly as the communications time decreases. Notice as well that robots further down the chain experience delayed errors corresponding to large δ, but not nearly as



(a)



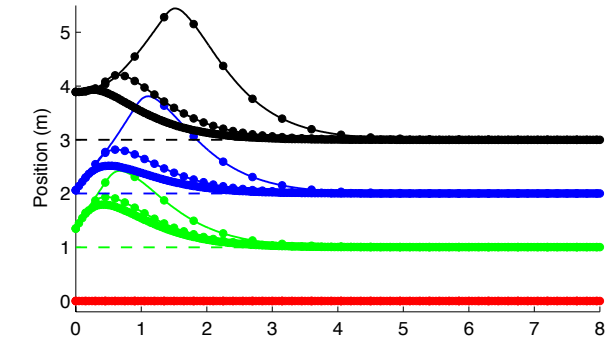
(b)

Figure 3.3: Simulation of a team of four agents with accelerating leader and expanding formation. Figure 3.3(a) shows the positions of each agent as solid lines and the dotted lines represent the desired position of each agent relative to the leader (This is only provided as visual reference. The actual error values are defined in (3.2.1)). Figure 3.3(b) illustrates that the error of each robot converges to zero.

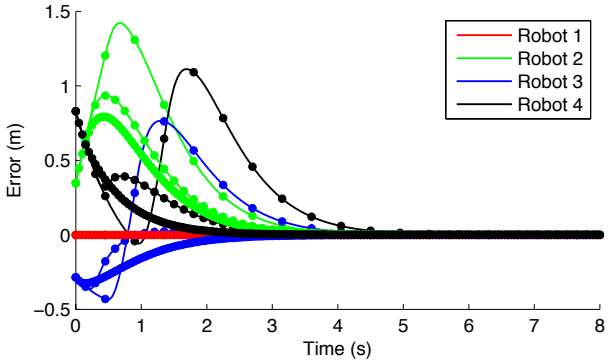
prominent with smaller δ. This simulation supports the conclusions made in Sect. 3.3 about the effects of communications time.

3.5 Experimental Results

This section experimentally evaluates the method presented in this chapter with three different studies. The first investigation in Sect. 3.5.1 analyzes trajectory tracking performance as desired velocities and accelerations vary. The second study in Sect. 3.5.2 tests



(a)



(b)

Figure 3.4: Simulations of a team of four agents with constant shape and a stationary leader, but with varying communications time $\delta = \{0.45, 0.15, 0.05\}$. The circles are displayed at instances of communications. Dotted lines are provided in Fig. 3.4(a) to show the desired position for each agent relative to the leader.

system performance given emulated communication degradation. Section 3.5.3 demonstrates system resilience to critical individual robot failure.

All robot are homogenous in these experimental trials per assumption (A2). However, (A1) is removed for all experimental trials since inequality constraints are required to ensure the robots' trajectories are feasible and within the workspace. By removing this assumption, no analytical solution to the QP exists. Instead, an off the shelf QP solver is be used instead. Assumption (A3) is used for trials in Sect. 3.5.1, but synchronous

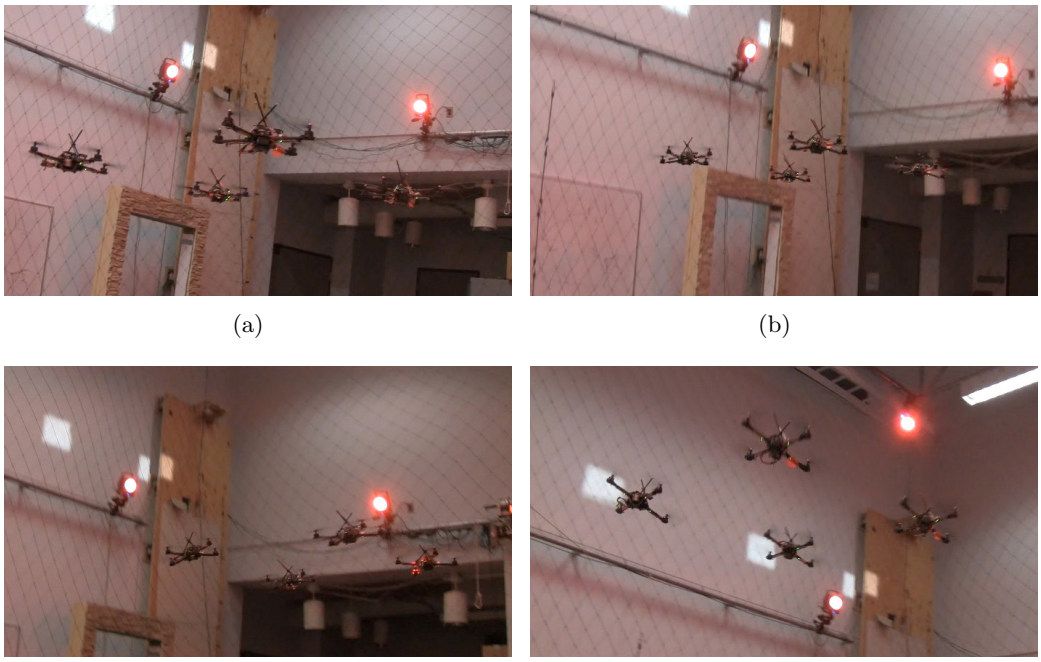
updates are necessarily removed in degraded network studies. To simulate real conditions in Sect. 3.5.3, the broken robot does not communicate with any robots after the simulated failure.

The experiments emulate true decentralized control by separately encapsulating the computation of each robot's trajectory with no global knowledge. The only information available to a robot is through emulated inter-robot communications. Individual messages are subjected to random time delays and dropped packets to simulate imperfect communications in Sect. 3.5.2.

All experimental studies consist of a formation of four robots with specified constant shape vectors. Robots utilize inequality constraints to enforce velocity and acceleration bounds. Dependence matrices ($\mathbf{D}$) are chosen such that each follower robot places equal weight on trajectories of all other robots with which it had recent communication.

3.5.1 Performance with Variable Maximum Velocity and Acceleration

To evaluate the performance of these methods with demanding trajectories, the leader trajectory is designed to give a variety of motions with high speeds and high accelerations in all dimensions. Time scaling of trajectories [53] is used to vary the velocities and accelerations demanded of the quadrotors. The duration of leader trajectories are 32, 24, 16, 12, and 10 sec. See Fig. 3.6(a) for a plot of tracked trajectories by the quadrotors for the trial requiring 16 s. To understand how well each robot is tracking its desired trajectory, see Fig. 3.6(b) for the cumulative distribution function of the error in $\mathbb{R}^3$ for the lead robot. All follower robots have similar errors to the lead robot for every trajectory tracked in every trial run. Figure 3.5 displays snapshots of a fast trajectory



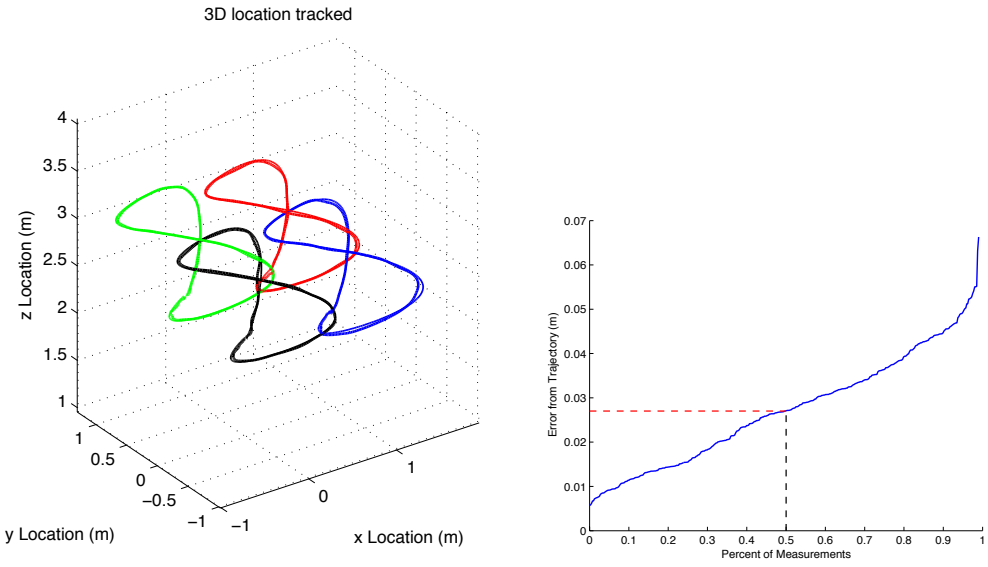
(a) (b)

(c) (d)

Figure 3.5: A team of four Ascending Technologies [5] Hummingbird quadrotors fly in formation while achieving maximum velocities and accelerations exceeding 3 $\frac{m}{s}$ and 6 $\frac{m}{s^2}$, respectively.

(with accelerations of greater than 0.5 g) being tracked highlighting that even during large deviation from near hover operation, all robots remain in close formation.

As the speeds of the trajectories are increased, it is expected that error from the desired trajectory should increase as time delays and modeling errors of quadrotor parameters become more significant. Datasets from all trials in this performance study were merged to analyze how well the robots maintained formation in demanding situations. Every data point collected was analyzed to relate speed and magnitude of desired acceleration to error from desired trajectory and error from formation with all other robots. The mean and variance of this data are plotted in Fig. 3.7. As one might expect, it is clear that as speed and acceleration increase, the error from the desired trajectory increases



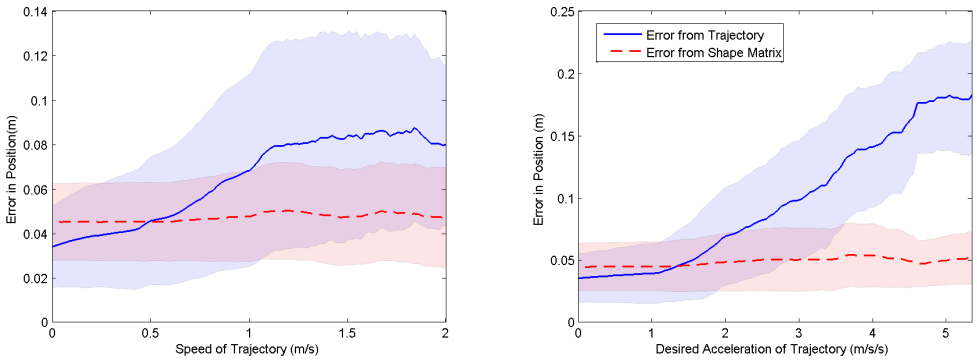
(a) (b)

Figure 3.6: The trajectories traversed by each robot in the formation where thin lines show the desired trajectory and bold lines indicate the actual trajectory (Fig. 3.6(a)). Note that the desired and actual trajectories are sufficiently close that only the actual trajectory is readily visible. The inverse cumulative probability distribution function for the error in $\mathbb{R}^3$ for the lead robot while tracking the waypoints used for the performance study (Fig. 5.14). Half of all measurements are within 2.7 cm of the desired trajectory.

substantially. However, the shape error remains relatively constant across all tested speeds and accelerations. This resilience to relative formation error during aggressive maneuvers confirms that these methodologies are sound and will ensure collision avoidance of robots in the formation.

3.5.2 Performance with Degraded Network Communication

This study is concerned with the practical complication of communication and sensing latencies, errors, and failures. To study how packet loss affects formation errors, various levels of failure of interagent communication are simulated and the formation response is measured. Each robot to robot communication is allowed to be transmitted successfully



(a) (b)

Figure 3.7: As the robots' speed and acceleration increase, the shape errors (deviations from the entries in the specified shape vectors) remain relatively constant. Each plot depicts the mean squared error (solid and dashed lines) and standard deviation in error (filled regions) between the desired and actual robot trajectories (solid, blue) and shape (dashed, red) using data collected from many trials.

with probability ρ. For example, a value of $\rho = 0.75$ signifies on average three out of every four attempted communications is successful. A large initial error trajectory was utilized to clearly demonstrate the decay to the desired shape. Figure 3.8 shows trials with four values of ρ ranging from 0.4 to 1.0. The slight initial offsets between trials are a result of different starting conditions between trials. As expected, the same decay rate is experienced regardless of the rate of communications failure. These results confirm that for even very unreliable networks, these controllers will converge to the desired shape.

3.5.3 Performance with Critical Robot Failures

To simulate a complete robot failure, at a prescribed time t_s, robot 2 ceases all communications and safely moves away from the remaining formation. The remaining agents initially attribute the lack of new information from robot 2 to a temporary communication failure and continue planning based on the most recent trajectory transmitted by

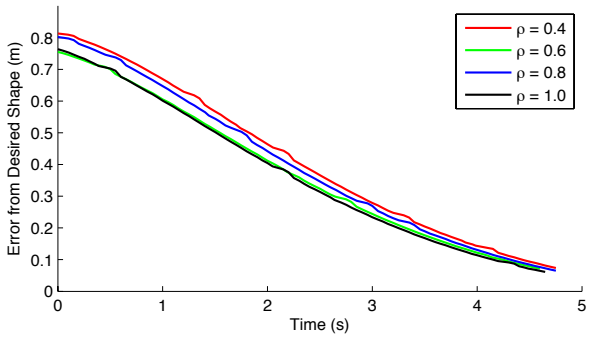


Figure 3.8: Convergence of robots to the desired shape with simulated communication failures. ρ is the probability that communications are successful.

robot 2. Over time, the remaining robots recognize that the data from robot 2 is too outdated to continue planning with and update their respective entries in the dependence matrix to reflect this fact. In practice, once the most recent communication with a robot is older than $t_h/2$, the dependence in that robot is set to zero and other dependence scores are scaled to preserve the row sum of $\mathbf{D}$. The same trial was also conducted with all robots functional throughout for comparison. The complete robot failure and baseline non-failure cases are compared in Fig. 3.9. The interagent distances between all remaining robots continue to decay at the same rate whether or not there is a robot failure. As long as failing robots do not physically disable other robots in a crash, this controller is designed to recognize the failure and automatically compensate without any formation tracking quality loss.

3.6 Chapter Summary

This chapter considers the problem of controlling a team of micro-aerial vehicles moving quickly through a three-dimensional environment while accurately maintaining a tight formation. The solution presented is based on the leader-follower control paradigm with

the follower robots controlling to maintain a desired formation shape. A set of potentially time varying shape vectors describe the desired ensemble shape and the dependence matrix dictates each robot's reliance on other the robots' state information for its trajectory generation and control. The computational complexity is independent of the total size of the group since each robot uses only local information to compute a trajectory. The method presented ensures that the system converges to the desired formation while undergoing bulk motion given sufficient communications rate. Unlike previous research on leader-follower control schemes or consensus algorithms, this approach addresses the synthesis of trajectories for multiple robots satisfying dynamic constraints and consensus across trajectories. An experimental evaluation of the approach on a team of quadrotors suggests that suitable performance is maintained as the group motions become increasingly aggressive even as inter-robot communication degrades.

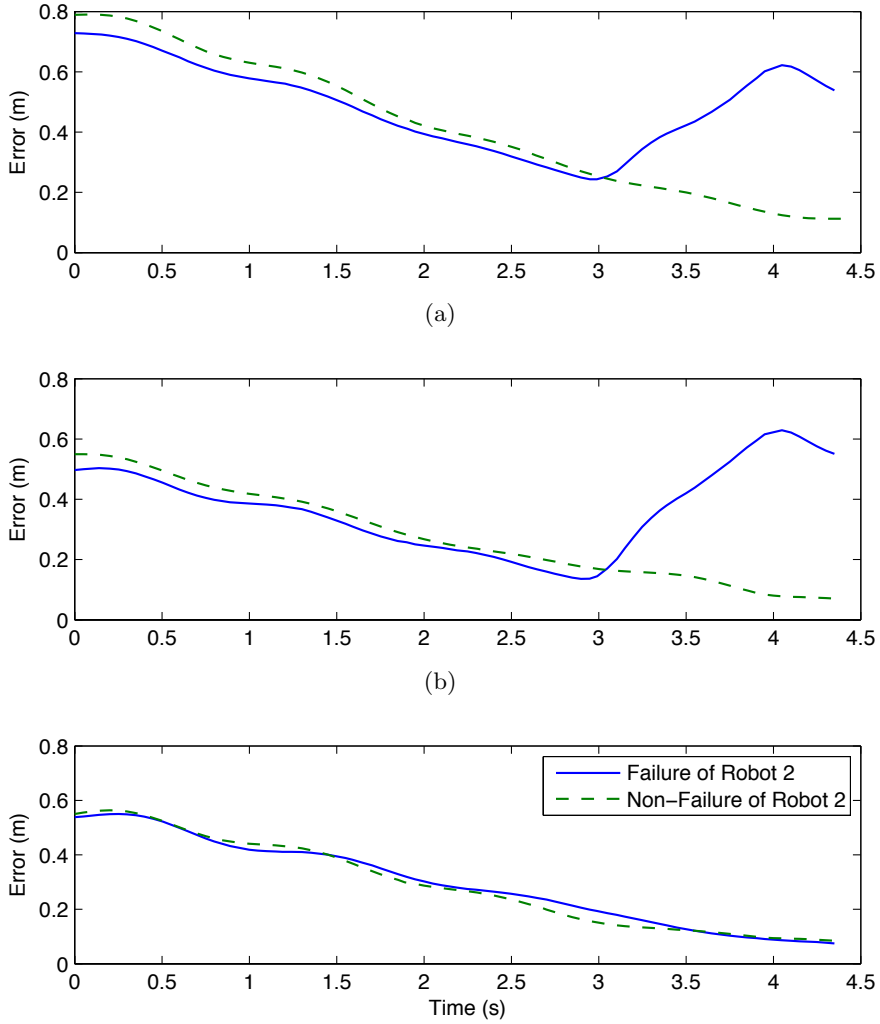


Figure 3.9: A simulated failure case of robot 2 at $t_s = 3$ sec with non-trivial starting error from the desired shape. The pairwise shape error between robots $\{1, 2\}$, $\{2, 3\}$, and $\{1, 3\}$ are shown in Figs. 3.9(a)–3.9(c), respectively.

Chapter 4: Concurrent Assignment and Planning of Trajectories CAPT

This chapter considers the problem of Concurrent Assignment and Planning of Trajectories (denoted CAPT) for a team of robots. This problem involves simultaneously addressing two challenges: (1) the combinatorially complex problem of finding a suitable assignment of robots to goal locations, and (2) the generation of collision-free, time parameterized trajectories for every robot. See Sect. 2.2 for a review of the challenges associated with planning for a team of robots. Additionally, Sect. 2.4 discusses the benefits of considering interchangeable robots. This chapter examines the CAPT problem for unlabeled (interchangeable) robots and proposes algorithmic solutions to two variations of the CAPT problem. The first algorithm, C-CAPT, is a provably-correct, complete, centralized algorithm which guarantees collision-free optimal solutions to the CAPT problem in an obstacle-free environment. To achieve these strong claims, C-CAPT exploits the synergy obtained by combining the two subproblems of assignment and trajectory generation to provide computationally-tractable solutions for large numbers of robots. This chapter then presents a decentralized solution to the CAPT problem through D-CAPT, a decentralized algorithm that provides suboptimal results compared to C-CAPT. These algorithms and resulting performance are demonstrated through simulation and experimentation.

By definition, the CAPT problem seeks an assignment of N unlabeled (permutation

invariant) robots to M desired goal locations. To explore the complexity of this problem, consider the case when $N = M$. Each possible assignment of robots to goals can be represented by an $N \times N$ permutation matrix and the space of all possible assignments is isomorphic to $\mathbb{S}^N$, or the group of all permutations. To completely enumerate all possible assignments requires $N!$ evaluations and is infeasible for any system with more than just a few robots. Fortunately, the task assignment problem occurs naturally in a number of disciplines including distributed computing, operations research, and robotics. As such, there has been significant study into generating optimal solutions and useful suboptimal solutions. While there are multiple classes of the task assignment problem, this work focuses on the linear assignment problem. The linear assignment problem seeks to minimize the sum of individual costs for robot-goal pairs. The well-known Hungarian Algorithm [41] solves the linear assignment problem with computational complexity bounded polynomially in the number of robots, $\mathcal{O}(N^3)$, and is the most efficient known method to optimally solve the linear task assignment problem. There are also a number of relaxations of the linear assignment problem to find near optimal solutions. For example [73] uses a heuristic to minimize the sum of Euclidean distance traveled for a simplified Euclidean linear assignment problem with complexity bound $\mathcal{O}(N^{\frac{5}{6}})$.

An approach to minimizing computational complexity is to consider decentralized solutions to the CAPT problem. The trajectory planning problem is naturally decoupled when robots are far away from each. Thus, it is logical to have robots independently plan trajectories to assigned goals, but reevaluate their assignments and planned trajectories as they approach each other. In this context, it is useful to introduce the abstraction of a communication or sensing range, h, outside which robots ignore their neighbors allowing

decentralized planning and execution. When a robot comes within h of a neighbor, it must coordinate its actions with its neighbor. Indeed it is possible to define reactive controllers using artificial potential functions [100] that modify trajectories locally when robots come close to each other while pursuing their assigned goals. In addition, there is literature on controlling groups of robots to goal destinations [59] or goal sets [10]. In general, these gradient descent approaches lead to unpredictable trajectories, may take a long time to converge, and some do not guarantee complete assignment.

The complete decentralized solution, D-CAPT, does not, in general, guarantee optimal solutions. However, every local modification of trajectories is associated with a reassignment of goals that improves the quality of the resulting solution. By considering only local interactions, D-CAPT has substantially reduced computational requirements. This allows the algorithm to scale well as the team size increase and suffers only a minor decrease in optimality.

After the presentation of preliminaries in Sect. 4.1, Sect. 4.2 introduces C-CAPT. This centralized optimal solution to the CAPT problem requires an obstacle-free environments and is computationally tractable for many hundreds of agents and goals. A basic iterative approach to visiting more goals than robots available is presented in 4.3. Not that this is similar to the Vehicle Routing Problem discussed in Sect. 2.5 and will be considered in more detail in Chapter 6. Section 4.4 demonstrates the flexibility of C-CAPT by incorporating the dynamics of a quadrotor UAV. Then Sect. 4.5 introduces D-CAPT, a distributed suboptimal version of C-CAPT. Finally, section 4.6 presents numerical simulations and analysis of C-CAPT and D-CAPT.

The work in this chapter was originally presented in [85, 87].

4.1 Preliminaries

Consider N robots with radius R navigating from initial locations to M desired goal locations in an n-dimensional Euclidean space. Define the set of integers between 1 and positive integer Z as: $\mathcal{I}_Z \equiv \{1, 2, \ldots, Z\}$. For example, if there are 3 goal locations, $M = 3$ and $\mathcal{I}_M = \{1, 2, 3\}$.

The location of the i^{th} robot is specified by $\mathbf{x}_i(t) \in \mathbb{R}^n$, $i \in \mathcal{I}_N$ and similarly, the j^{th} goal location is specified by $\mathbf{g}_j \in \mathbb{R}^n$, $j \in \mathcal{I}_M$. Next, define the Nn-dimensional system state vector, $\mathbf{X} \in \mathbb{R}^{Nn}$:

$$\mathbf{X}(t) = [\mathbf{x}_1(t)^{\mathrm{T}}, \mathbf{x}_2(t)^{\mathrm{T}}, \ldots, \mathbf{x}_N(t)^{\mathrm{T}}]^{\mathrm{T}}.$$

and similarly define the system goal state vector $\mathbf{G} \in \mathbb{R}^{Mn}$:

$$\mathbf{G} = [{\mathbf{g}_1}^{\mathrm{T}}, {\mathbf{g}_1}^{\mathrm{T}}, \ldots, {\mathbf{g}_M}^{\mathrm{T}}]^{\mathrm{T}}$$

Define the *assignment matrix* $\phi \in \mathbb{R}^{N \times M}$, which assigns agents to goals:

$$\phi_{ij} = \begin{cases} 1 & \text{if robot } i \text{ is assigned to goal } j \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.1.1}$$

It is required that either all goals to be assigned or all robots to be assigned which results in:

$$\begin{aligned} \phi^{\mathrm{T}}\phi &= I_M \qquad \text{if } N \geq M \\ \phi\phi^{\mathrm{T}} &= I_N \qquad \text{if } N \leq M \end{aligned} \tag{4.1.2}$$

where I_Z is the $Z \times Z$ identity matrix. It is also useful to define the expanded assignment matrix $\Phi \equiv \phi \otimes I_n$ where $\otimes$ signifies the Kronecker product.

The CAPT problem seeks to find Nn-dimensional trajectories:

$$\gamma : t \to \mathbf{X}$$

where t_0 and t_f are the initial and final times respectively. Initial robot positions $\mathbf{x}_i(t_0)\ \forall i \in \mathcal{I}_N$ provide initial conditions for the trajectories:

$$\gamma(t_0) = \mathbf{X}(t_0) \tag{4.1.3}$$

The assignment is defined such that it provides terminal conditions such that goal locations are occupied by robots:

$$\Phi^{\mathrm{T}}\gamma(t_f) = \mathbf{G}$$

However, in the case that $M > N$, these assignments are not unique. To find a unique assignment, premultiply this condition by Φ:

$$\Phi\Phi^{\mathrm{T}}\gamma(t_f) = \Phi\mathbf{G}$$

and solve for $\gamma(t_f)$:

$$\gamma(t_f) = (\Phi\Phi^{\mathrm{T}})^{-1}\Phi\mathbf{G}$$

Using (4.1.2), $M > N$ implies that $\Phi\Phi^{\mathrm{T}} = I_{Nn}$. Therefore, the assignment of robots to goals supplies the following trajectory terminal conditions:

$$\begin{aligned} \Phi^{\mathrm{T}}\gamma(t_f) &= \mathbf{G} \qquad \text{if } N \geq M \\ \gamma(t_f) &= \Phi\mathbf{G} \qquad \text{if } N < M \end{aligned} \tag{4.1.4}$$

Clearance ϵ is defined as the minimum space between any pair of robots at any time during the trajectory:

$$\epsilon(t) = \inf_{i \neq j \in \mathcal{I}_N, t \in [t_0, t_f]} ||\mathbf{x}_i(t) - \mathbf{x}_j(t)|| - 2R$$

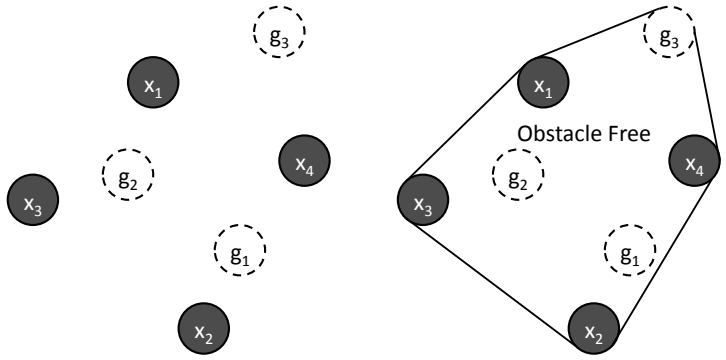


Figure 4.1: For the locations of goals and agents in $\mathbb{R}^2$ in Fig. 4.1(a), $\mathcal{K}$ is designated by the closed area in Fig. 4.1(b).

With the exception of Sect. 4.4, the robots are assumed to have simple first order dynamics:

$$\begin{aligned} \dot{\mathbf{x}}_i(t) &= \mathbf{u}_i \\ ||\mathbf{u}_i||_2 &\leq v_{max} \end{aligned} \tag{4.1.5}$$

To ensure collision avoidance for all robots, clearance is required to always be greater than zero:

$$\epsilon(t) > 0 \quad t \in [t_0, t_f] \tag{4.1.6}$$

Define $\mathcal{K}$ as the convex hull of initial locations and goal locations with the Minkowski sum of a ball of radius R:

$$\mathcal{K} \equiv \text{conv}\left(\{\mathbf{x}_i(t_0)\big| i \in \mathcal{I}_N\} \cup \{\mathbf{g}_j\big| j \in \mathcal{I}_M\}\right) \oplus \mathcal{B}_R \tag{4.1.7}$$

See Fig. 4.1 for a 2-dimensional pictorial example of $\mathcal{K}$. Section 4.2 presents C-CAPT, which requires an obstacle-free environment such that $\mathcal{K}$ is free of obstacles.

4.2 c-Capt: a Centralized Obstacle-Free Capt Solution

This section explicitly defines the requirements of a centralized Capt solution and the necessary assumptions for an obstacle-free environment. Then, a minimum distance solution to the assignment problem in Sect. 4.2.1 is studied. Section 4.2.2 modifies this cost function and evaluates the solution generated by a minimum velocity squared trajectory. Finally, Sect. 4.2.3 presents equations which generate a solution to the Capt problem in an obstacle-free environment.

Problem Definition c-Capt seeks to find optimal trajectories $\gamma^\star(t)$ which minimize a specified cost functional:

$$\begin{aligned} \gamma^\star(t) = \underset{\gamma(t)}{\operatorname{argmin}} \quad & \int_{t_0}^{t_f} L(\gamma(t))dt \\ \text{subject to} \quad & \text{Eq. (4.1.1) : Valid Assignment} \\ & \text{Eq. (4.1.2) : Full Resource Utilization} \\ & \text{Eq. (4.1.3) : Initial Conditions} \\ & \text{Eq. (4.1.4) : Terminal Conditions} \\ & \text{Eq. (4.1.5) : Robot Capabilities} \\ & \text{Eq. (4.1.6) : Collision Avoidance} \end{aligned} \tag{4.2.1}$$

Assumptions Explicitly stating the assumptions required for c-Capt:

(A1) All robots are interchangeable with no preference of goal location.

(A2) The extend of each robot is confined to $\mathcal{B}_R$, a ball with radius R.

(A3) The region $\mathcal{K}$ as defined in (4.1.7) is obstacle-free.

(A4) The initial and goal locations are spaced Δ apart:

$$\begin{aligned} ||\mathbf{x}_i(t_0) - \mathbf{x}_j(t_0)|| > \Delta \quad \forall\ i \neq j \in \mathcal{I}_N, \\ ||\mathbf{g}_i - \mathbf{g}_j|| > \Delta \quad \forall\ i \neq j \in \mathcal{I}_M \end{aligned} \tag{4.2.2}$$

where Δ will be defined later. Clearly $\Delta > 2R$ to ensure collision avoidance (4.1.6) at the boundary conditions. Additionally require:

$$||\mathbf{x}_i(t_0) - \mathbf{g}_j|| > \Delta \quad \forall i \in \mathcal{I}_N, j \in \mathcal{I}_M \quad \text{if} \quad N > M$$

(A5) Robots are fully actuated differentially flat systems, do not have any actuation error, and always have perfect state knowledge.

Sections 4.2.1 and 4.2.2 propose different cost functions $L(\gamma)$ in (4.2.1) which both seek to simultaneously find an assignment matrix ϕ and collision-free trajectories $\gamma(t)$ for all robots in the system such that the boundary conditions (4.1.3, 4.1.4) are satisfied.

4.2.1 Minimum Sum of Distances Trajectories

The first cost function analyzed is commonly applied to the location assignment problem and seeks to minimize the sum of distances traveled by all agents:

$$\begin{aligned} \underset{\phi,\gamma(t)}{\text{minimize}} \quad & \sum_{i=1}^{N} \int_{t_0}^{t_f} \sqrt{\dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t)} dt \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.5), (4.1.6) \end{aligned}$$

Temporarily ignoring clearance requirements, it is clear that the solution reduces to positive progress on straight line paths for $t \in [t_0, t_f]$. Thus, this problem reduces to:

$$\begin{aligned} \underset{\phi}{\text{minimize}} \quad & \sum_{j=1}^{M} \sum_{i=1}^{N} \phi_{ij} \left|\left|\mathbf{x}_i(t_0) - \mathbf{g}_j\right|\right|_2 \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.5) \end{aligned} \tag{4.2.3}$$

This is the well-known transportation assignment problem. Theorem 4.2.1 demonstrates that the assignment from this optimization guarantees paths that almost never intersect in any $n > 1$-dimensional Euclidean space.

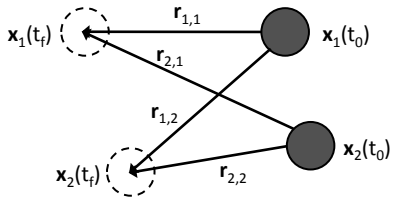


Figure 4.2: For agents in n-dimensional Euclidean space, if paths intersect and are not collinear as is the case for straight lines along $\mathbf{r}_{1,2}$ and $\mathbf{r}_{2,1}$, it is clear that by using the triangle inequality, switching assignments will always lead to shorter non-intersecting paths.

Theorem 4.2.1. *The optimal assignment ϕ using the minimum sum of distance optimization in* (4.2.3) *results in non-intersecting paths with the exception of the special case when a pair of robots have collinear start and goal locations.*

Proof. Assume the paths of agents i and j intersect but do not all fall on a line. These paths necessarily exist in a plane and therefore can be reduced to an equivalent $n = 2$ problem. Since these paths intersect, the switching the assignment will always reduce the sum of distances by using the triangle inequality and the given ϕ is not optimal for (4.2.3). Therefore, the minimum assignment found in (4.2.3) will never result in intersecting paths. □

Unfortunately, it can be shown with the simple example depicted in Fig. 4.3 that agents with finite extent using the assignment from (4.2.3) are not guaranteed collision-free trajectories, rendering these trajectories useless for real robots with physical extent. This minimum distance assignment can be used to find trajectories which avoid collisions, but are suboptimal, may require enlargement of the region $\mathcal{K}$, and are more difficult to compute than those which will be presented in Sect. 4.2.2.

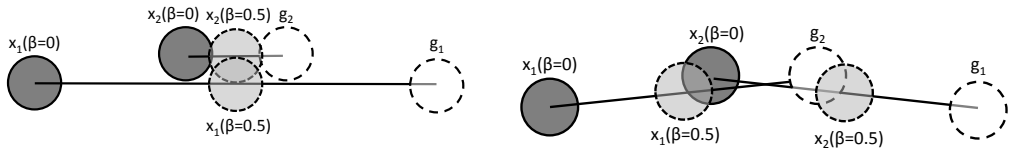


Figure 4.3: For the example with two agents in Fig. 4.3(a) it is clear that the minimum sum of distances paths (calculated by (4.2.3)) never intersect. However, having intersection-free paths does not guarantee collision-free trajectories for agents with finite size. In this case, merely switching goal assignments, as shown in Fig. 4.3(b), does ensure collision-free trajectories. It should be noted that minimizing the sum of distance traveled squared arrives at the collision-free assignment in Fig. 4.3(b).

4.2.2 Minimum Velocity Squared Trajectories

The second proposed method is to minimize the sum of the integral of velocity squared traveled by all agents:

$$\begin{aligned} \underset{\phi,\gamma(t)}{\text{minimize}} \quad & \sum_{i=1}^{N} \int_{t_0}^{t_f} \dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t) dt \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.5), (4.1.6) \end{aligned}$$

which is equivalent to:

$$\begin{aligned} \underset{\phi,\gamma(t)}{\text{minimize}} \quad & \int_{t_0}^{t_f} \dot{\mathbf{X}}(t)^{\mathrm{T}} \dot{\mathbf{X}}(t) dt \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.5), (4.1.6) \end{aligned} \tag{4.2.4}$$

The remainder of this section details C-CAPT and its use as a solution to this problem

To clarify how the optimization in Sect. 4.2.2 differs from that in Sect. 4.2.1, consider moving a contiguous block of a number of books each with identical width to another contiguous block, but moved one book over and ignoring collisions. One solution is to move the first book to the last position, where another is to move each book one position over. Both schemes result in the same sum of distance traveled, however moving each book

one unit over results in a lower sum of distances squared as a result of distance squared being a strictly convex cost function. Notice that in the many smaller moves solution, one book will not cross another. To relate this simple example to the CAPT problem, note that all of the books can be simultaneously shifted to their new location without collision.

Temporarily relax (4.2.4) to ignore the clearance requirements in (4.1.6):

$$\begin{aligned} \underset{\phi,\gamma(t)}{\text{minimize}} \quad & \int_{t_0}^{t_f} \dot{\mathbf{X}}(t)^{\mathrm{T}} \dot{\mathbf{X}}(t) dt \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.5) \end{aligned} \tag{4.2.5}$$

The solution to (4.2.5) will consist of straight line trajectories which satisfy the boundary conditions while minimizing the sum of distance traveled squared.

Optimal Assignment

First consider the assignment problem and create a distance squared matrix $D \in \mathbb{R}^{N \times M}$:

$$D_{ij} = ||\mathbf{x}_i(t_0) - \mathbf{g}_j||^2 \qquad i \in \mathcal{I}_N, j \in \mathcal{I}_M$$

Then solve for the optimal distance squared assignment matrix $\phi^\star$:

$$\phi^\star = \underset{\phi}{\text{argmin}} \quad \sum_{i=1}^{N} \sum_{j=1}^{M} \phi_{ij} D_{ij} \tag{4.2.6}$$

Note that $\phi^\star$ satisfies (4.1.1) and (4.1.2).

This is a linear assignment problem and therefore an optimal assignment solving algorithm such as the Hungarian Algorithm can be used to solve for $\phi^\star$. In MATLAB on a standard laptop computer, solve times of under ten seconds for 400 robot/goal pairs is typical. Figure 4.4 displays run times that trend toward cubic growth in N.

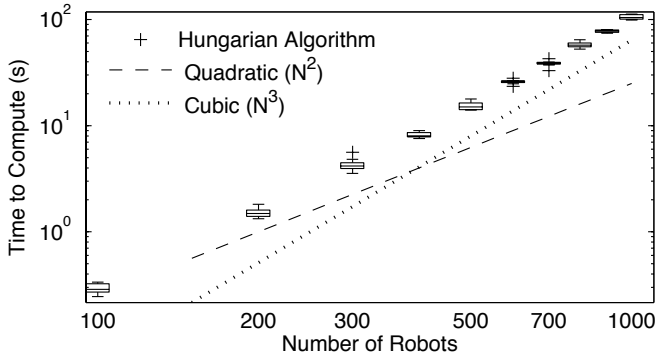


Figure 4.4: Solution times for the Hungarian Algorithm to solve (4.2.6) in MATLAB on a standard laptop computer. The box-plot is generated based on 10 trials for each value of N. Note that the algorithm runtime trends toward N^3 growth. Boxes represent the 25th and 75th percentiles, whiskers denote the 99% confidence interval and outliers are marked using "+".

Trajectory Generation

The termination time t_f can be computed as follows:

$$t_f = \underset{i}{\text{maximize}} \quad \frac{\left|\left|\mathbf{x}_i(t_0) - \sum_{j=1}^{M} \phi_{ij}^{\star}\mathbf{g}_j\right|\right|_2}{v_{max}} \tag{4.2.7}$$

Define the mapping $\beta : t \to [0, 1]$ as a polynomial function of time:

$$\beta(t) \equiv \sum_{i=0}^{k} \alpha_i t^i$$

such that $\beta(t_0) = 0$ and $\beta(t_f) = 1$.

A straight forward application of the calculus of variations shows that the trajectories which minimize the integral of velocity squared are those with constant velocity and satisfy the boundary conditions:

$$\gamma^{\star}(t) = (1 - \beta(t))\,\mathbf{X}(t_0) + \beta(t)\left(\Phi\mathbf{G} + (I_{Nn} - \Phi\Phi^{\mathrm{T}})\mathbf{X}(t_0)\right) \tag{4.2.8}$$

where β is defined by the first order polynomial:

$$\alpha_0 = \frac{-t_0}{t_f - t_0}, \quad \alpha_1 = \frac{1}{t_f - t_0}, \quad \alpha_2 = 0 \ \ldots \ \alpha_k = 0$$

It is clear that at $t = t_0$, $\gamma^\star(t_0) = \mathbf{X}(t_0)$, and therefore (4.2.8) satisfies the initial conditions in (4.1.3).

In equation 4.2.8, the term, $I_{Nn} - \Phi\Phi^{\mathrm{T}}$, selects all unassigned robots and ensures that these robots remain at their original location. If $M \geq N$, using (4.1.2), this term will disappear as expected, resulting from the fact that all robots will be assigned.

Verify that $\gamma^\star(t)$ satisfies the final boundary conditions specified in (4.1.4):

$$\gamma^\star(t_f) = \Phi\mathbf{G} + (I_{Nn} - \Phi\Phi^{\mathrm{T}})\mathbf{X}(t_0)$$

If $M \geq N$, then using (4.1.2), $\Phi\Phi^{\mathrm{T}} = I_{Nn}$ and $\gamma^\star(t_f) = \Phi\mathbf{G}$. If instead $N \geq M$, premultiply (4.2.8) by Φ^{T}:

$$\Phi^{\mathrm{T}}\gamma^\star(t_f) = \Phi^{\mathrm{T}}\Phi\mathbf{G} + (\Phi^{\mathrm{T}}I_{Nn} - \Phi^{\mathrm{T}}\Phi\Phi^{\mathrm{T}})\mathbf{X}(t_0)$$

From (4.1.2), $\Phi^{\mathrm{T}}\Phi = I_{Mn}$ to thus verify that $\Phi^{\mathrm{T}}\gamma^\star(t_f) = \mathbf{G}$. Therefore, the trajectories defined in (4.2.8) satisfy the terminal conditions in (4.1.4).

The definition of the termination time t_f in (4.2.7) guarantees that all robots satisfy their actuation bounds in (4.1.5).

Collision Avoidance

Thus far, it has been shown that (4.2.6) and (4.2.8) generate the solution to the relaxed problem without considering collisions in (4.2.5). However, utilizing the properties of the CAPT problem demonstrates in Theorem 4.2.3 that if $\Delta > 2\sqrt{2}R$, these equations also provide the solution to the full collision avoidance problem in (4.2.4).

For notational convenience, define:

$$\mathbf{r}_{ij} \equiv \mathbf{x}_j(t_f) - \mathbf{x}_i(t_0) \qquad \mathbf{u}_{ij} \equiv \mathbf{x}_j(t_0) - \mathbf{x}_i(t_0) \qquad \mathbf{w}_{ij} \equiv \mathbf{x}_j(t_f) - \mathbf{x}_i(t_f)$$

Lemma 4.2.2. *The optimal solutions to* (4.2.5) *satisfy:*

$$\mathbf{w}_{ij}{}^{T}\mathbf{u}_{ij} \geq 0 \ \forall i,j \in \mathcal{I}_N \tag{4.2.9}$$

Proof. Globally minimize the sum of integrated velocity squared in (4.2.5) such that switching goal states of agent i with agent j will not decrease the sum of distance squared, or:

$$||\mathbf{r}_{ii}||^2 + ||\mathbf{r}_{jj}||^2 \leq ||\mathbf{r}_{ij}||^2 + ||\mathbf{r}_{ji}||^2 \ \forall i,j \in \mathcal{I}_N \tag{4.2.10}$$

Then substitute:

$$||\mathbf{r}_{ij}||^2 = \mathbf{r}_{ij}{}^{\mathrm{T}}\mathbf{r}_{ij} = \mathbf{x}_j(t_f)^{\mathrm{T}}\mathbf{x}_j(t_f) - 2\mathbf{x}_i(t_0)^{\mathrm{T}}\mathbf{x}_j(t_f) + \mathbf{x}_i(t_0)^{\mathrm{T}}\mathbf{x}_i(t_0)$$

into (4.2.10) and simplify:

$$(\mathbf{x}_j(t_f) - \mathbf{x}_i(t_f))^{\mathrm{T}}(\mathbf{x}_j(t_0) - \mathbf{x}_i(t_0)) \geq 0 \ \forall i,j \in \mathcal{I}_N$$

or

$$\mathbf{w}_{ij}{}^{\mathrm{T}}\mathbf{u}_{ij} \geq 0 \ \forall i,j \in \mathcal{I}_N \tag{4.2.11}$$

□

Theorem 4.2.3. *If* $\Delta > 2\sqrt{2}R$*, trajectories in* (4.2.8) *will satisfy* (4.1.6) *and be collision-free.*

Proof. The location of robot i following the trajectory specified in (4.2.8) is:

$$\mathbf{x}_i(t) = (1-\beta)\mathbf{x}_i(t_0) + \beta\mathbf{x}_i(t_f)$$

Therefore the distance between robots i and j is:

$$||\mathbf{x}_j(t) - \mathbf{x}_i(t)|| = ||(1-\beta)\mathbf{x}_j(t_0) + \beta\mathbf{x}_j(t_f) - (1-\beta)\mathbf{x}_i(t_0) - \beta\mathbf{x}_i(t_f)||$$

and the distance squared between these robots is:

$$\begin{aligned} ||\mathbf{x}_j - \mathbf{x}_i||^2 &= ||(1-\beta)(\mathbf{x}_j(t_0) - \mathbf{x}_i(t_0)) + \beta(\mathbf{x}_j(t_f) - \mathbf{x}_i(t_f))||^2 \\ &= ||\mathbf{u}_{ij} + \beta(\mathbf{w}_{ij} - \mathbf{u}_{ij})||^2 \end{aligned}$$

Rewrite this result:

$$||\mathbf{x}_j - \mathbf{x}_i||^2 = (\mathbf{u}_{ij} + \beta(\mathbf{w}_{ij} - \mathbf{u}_{ij}))^{\mathrm{T}}(\mathbf{u}_{ij} + \beta(\mathbf{w}_{ij} - \mathbf{u}_{ij})) \tag{4.2.12}$$

Define for notational convenience:

$$a \equiv {\mathbf{u}_{ij}}^{\mathrm{T}}\mathbf{u}_{ij} \qquad b \equiv {\mathbf{w}_{ij}}^{\mathrm{T}}\mathbf{u}_{ij} \qquad c \equiv {\mathbf{w}_{ij}}^{\mathrm{T}}\mathbf{w}_{ij}$$

Equation (4.2.12) simplifies to:

$$||\mathbf{x}_i - \mathbf{x}_j||^2 = a - 2\beta(a-b) + \beta^2(a - 2b + c) \tag{4.2.13}$$

Find the value of β which minimizes the distance squared (and therefore the distance) between agents i and j ($i \neq j$):

$$\beta_{ij}^{\star} = \underset{\beta}{\operatorname{argmin}} \; ||\mathbf{x}_i - \mathbf{x}_j|| = \frac{a-b}{a-2b+c} \tag{4.2.14}$$

If $\beta_{ij}^{\star}$ is outside the range $[0, 1]$, the agents are at a minimum at either the start or end of the trajectory and the agents will not collide due to **(A4)**. If however, $\beta_{ij}^{\star} \in [0, 1]$, the minimum distance between robots i and j over the course of their planned trajectories can be found by combining (4.2.13) and (4.2.14): The minimum distance squared is:

$$||\mathbf{x}_i - \mathbf{x}_j||_{\min}^2 = \frac{ac - b^2}{a - 2b + c} \tag{4.2.15}$$

Next, starting conditions which minimize this minimum distance can be analyzed. From their definitions, a, b, and c are all independent with the exception that $b < c$

and $b < a$ are guaranteed from the fact that $\beta \in [0, 1]$. It is clear that this minimum-minimum distance decreases as b decreases. However, as shown in Lemma 4.2.2, $b \geq 0$ and the minimum possible distance between robots occurs when $b = 0$:

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\min} \geq \frac{ac}{a+c}$$

Assumption **(A4)** specifies:

$$||\mathbf{x}_i(t_0) - \mathbf{x}_j(t_0)|| > \Delta \qquad\qquad ||\mathbf{x}_i(t_f) - \mathbf{x}_j(t_f)|| > \Delta$$

These are equivalent to:

$$\sqrt{\mathbf{u}_{ij}{}^{\mathrm{T}}\mathbf{u}_{ij}} > \Delta \qquad\qquad \sqrt{\mathbf{w}_{ij}{}^{\mathrm{T}}\mathbf{w}_{ij}} > \Delta$$

From the assumption $\Delta > 2\sqrt{2}R$, these inequalities are equivalent to:

$$a > 8R^2 \qquad\qquad c > 8R^2$$

Using these inequalities, it is straightforward to show:

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\min} \geq \frac{ac}{a+c} > 4R^2$$

And therefore:

$$||\mathbf{x}_i - \mathbf{x}_j||_{\min} > 2R$$

Thus a robot with radius R can never intersect another robot with radius R and collision avoidance is guaranteed. □

4.2.3 c-Capt Definition

c-Capt is defined as the solution to (4.2.4). Equation 4.2.16 reiterates (4.2.6) and (4.2.8), which provide collision-free trajectories that satisfy all requirements for the obstacle-free

CAPT problem.

$$\begin{aligned}\phi^\star &= \underset{\phi}{\text{argmin}} \quad \sum_{i=1}^{N}\sum_{j=1}^{M}\phi_{ij}D_{i,j} \\ \gamma^\star(t) &= (1-\beta(t))\,\mathbf{X}(t_0) + \beta(t)\left(\Phi^\star\mathbf{G} + (I_{Nn} - \Phi^\star\Phi^{\star\mathrm{T}})\mathbf{X}(t_0)\right)\end{aligned} \tag{4.2.16}$$

4.3 c-CAPT for Fewer Robots than Goals

In the event that $M > N$, or there are more goal locations to visit than robots to assign, there are many possible solutions that may be considered to solve the CAPT problem. One potential algorithm design is to use c-CAPT to find the set of trajectories which safely navigate the robots to the goals which minimizing the sum of distance traveled squared and consider the problem solved. This is appropriate when the mission goal is for each robot to be occupying one of the goal locations.

In the event that the operator wants the robots to inspect each of the $M \gg N$ goal locations, the problem could alternately be cast as an instance of the vehicle routing problem (VRP). This will be addressed in far greater detail in Chapter 6, but Algorithm 1 presents a simple solution to the VRP that guarantees all goals will be visited while ensuring collision avoidance and dynamic constraints by iteratively solving c-CAPT for assignments and trajectories until there are no remaining unreached goal locations. See Fig. 4.10 for a simulated example with 6 robots and 20 goal locations.

Algorithm 1 c-CAPT with $M > N$

```
while M > N do
    c-CAPT
    remove goals assigned by φ*
c-CAPT
```

While the paths generated by Algorithm 1 provide a solution to the VRP, the so-

lution is not necessarily optimal and in fact can be very suboptimal. Each iteration of C-CAPT guarantees local optimality in the sense of (4.2.4). For $M \approx N$, Algorithm 1 generates solutions very close to the optimal solution of the vehicle routing problem. As the ratio of goals to robots (M/N) increases, the benefits from solving the CAPT problem diminish and solutions from Algorithm 1 appear similar to a greedy solution of the traveling salesman problem.

4.4 C-CAPT for Dynamic Robots

As detailed in Appendix A, the dynamics of the quadrotor necessitate $\mathbb{C}^3$ smooth trajectories for accurate tracking. It is well known that minimizing the square of the norm of the snap (the fourth derivative of position) of the trajectory yields reference trajectories and inputs that are excellent for quadrotors [53]. Fortunately, this only results in a simple modification to the cost function in (4.2.4):

$$\begin{aligned} \underset{\phi,\gamma(t)}{\text{minimize}} \quad & \int_{t_0}^{t_f} \dddot{\mathbf{X}}(t)^{\mathrm{T}} \dddot{\mathbf{X}}(t) dt \\ \text{subject to} \quad & (4.1.1), (4.1.2), (4.1.3), (4.1.4), (4.1.6), \end{aligned} \tag{4.4.1}$$

A straight forward application of optimal control theory shows that each of the individual robot trajectories returned from this optimization will be those that minimize snap along straight line paths from the initial position to the goal location. It also follows that these minimum snap trajectories are seventh order polynomial functions of time. Because homogeneous boundary conditions are required (zero velocity, acceleration, and jerk at the initial and final conditions) the minimum snap trajectory has the same path as before but reparameterized by $\beta(t)$ with $k = 7$. This analysis can be easily applied to any robotic system. For example, a mobile ground robot which can be modeled as a second order

system would require a third order polynomial $\beta(t)$ with $k = 3$.

Because the boundary conditions are homogeneous, the integral of snap squared for a given trajectory will be exactly a constant factor times the integral of velocity squared in the original problem statement (4.2.16). This factor is constant for all assignments and therefore, the optimal assignment is independent of $\beta(t)$, and is the same as that obtained by solving the original problem.

4.5 D-CAPT: a Decentralized Algorithm for Obstacle-Free CAPT

For very large systems, solving the Hungarian Algorithm begins to be prohibitively expensive as seen in Fig. 4.4. This, coupled with the fact that interactions are inherently local, means that a decentralized algorithm may be preferred even when considering a fully connected network with centralized coordinator to improve computational performance at the cost of optimality.

This section exploits some properties of the centralized minimum velocity squared solution proposed in Sect. 4.2.2 to formulate a computationally tractable, online, decentralized algorithm to generate collision-free paths for all robots. The decentralized method takes inspiration from (4.2.11) and is based on reassignment of goal locations to arrive at a locally-optimal solution.

First, define the robots' communication or sensing range, h. When a robot comes within h of a neighbor, it must coordinate its actions with its neighbor.This distance h must be large enough so that robots can ignore neighbors outside this range.

This section requires assumptions **(A1)**-**(A5)** as well as two additional assumptions in **(A6)**-**(A7)**:

(A6) All robots are capable of exchanging information about their current state and their assigned goals to other robots closer than distance h. Of course h must be greater than Δ. If communication or sensing take significant time, it is clear that $h >> \Delta$.

(A7) $N = M$, where each goal location is initially assigned to exactly one robot.

As a result of robots communicating with neighboring robots within the communications range h, a moving robot can constantly be encountering new neighbors, and therefore learn new information about its neighbors, and by extension, information from its neighbors' neighbors and so on. The key feature of this algorithm is for every message sent, the system is locally minimizing a modified version of the cost functional in (4.2.4).

Similar to how communications were treated in Chapter 3, define the communication set $\mathcal{C}_i(t)$ as a list of all robots within the communications range of agent i at time t:

$$\mathcal{C}_i(t) = \{j \mid \|\mathbf{x}_j(t) - \mathbf{x}_i(t)\| \leq h, j \neq i\} \subset \mathcal{I}_N \tag{4.5.1}$$

Then, define the update list $\mathcal{U}_i(t) \subset \mathcal{C}_i(t)$ as the list of robots to which robot i will attempt to send new information.

Again, use t_c to denote the current time of computation such that $t_0 \leq t_c < t_f$. Slightly modify the definitions of $\mathbf{u}_{ij}$ and $\mathbf{r}_{ij}$:

$$\mathbf{r}_{ij} \equiv \mathbf{x}_j(t_f) - \mathbf{x}_i(t_c) \qquad \mathbf{u}_{ij} \equiv \mathbf{x}_j(t_c) - \mathbf{x}_i(t_c)$$

Redefine $\mathbf{g}_i$ as the goal currently assigned to robot i.

In D-CAPT presented in Algorithm 2, the i^{th} robot locally minimizes the contribution to (4.2.4) from every pair of i and j that satisfy (4.5.1):

$$\underset{\mathbf{g}_i, \mathbf{g}_j, \gamma(t)}{\text{minimize}} \quad \int_{t_c}^{t_f} \dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t) dt + \int_{t_c}^{t_f} \dot{\mathbf{x}}_j(t)^{\mathrm{T}} \dot{\mathbf{x}}_j(t) dt$$

The trajectory for the remaining time $t \in [t_c, t_f]$ is computed in a similar fashion to the centralized version (4.2.8):

$$\mathbf{x}_i(t) = \left(1 - \frac{t - t_c}{t_f - t_c}\right) \mathbf{x}_i(t_c) + \left(\frac{t - t_c}{t_f - t_c}\right) \mathbf{g}_i \tag{4.5.2}$$

Algorithm 2 $\mathbf{x}_i(t) =$ D-CAPT$(\mathbf{x}_i(t_0), \mathbf{g}_i)$

compute trajectory using (4.5.2)
$\mathcal{U}_i = \mathcal{C}_i(t_0)$
$t_{prev} \leftarrow t_0$
while $t < t_f$ **do**
  $t_c \leftarrow t$

  for $j \in \mathcal{C}_i(t_c)$ **do**
    if $j \notin \mathcal{C}_i(t_{\text{prev}})$ **then**
      $\mathcal{U}_i = \mathcal{U}_i \bigcup j$
  $\mathcal{U}_i = \mathcal{U}_i \bigcap \mathcal{C}_i(t_c)$

  for $j \in \mathcal{U}_i$ **do**
    request $\mathbf{x}_j(t_c)$ and $\mathbf{g}_j$ from agent j
    if $\mathbf{u}_{ij}{}^{\mathrm{T}} \mathbf{w}_{ij} < 0$ **then**
      send to robot j to change goal location to $\mathbf{g}_i$
      $\mathbf{g}_i \leftarrow \mathbf{g}_j$
      $\mathcal{U}_i = \mathcal{C}_i(t_c)$
      recompute trajectory using (4.5.2)
    $\mathcal{U}_i = \mathcal{U}_i \setminus j$

  if robot j sends $\mathbf{g}_j$ as the new robot i goal **then**
    $\mathbf{g}_i \leftarrow \mathbf{g}_j$
    $\mathcal{U}_i = \mathcal{C}_i(t_c)$
    recompute trajectory using (4.5.2)
    $\mathcal{U}_i = \mathcal{U}_i \setminus j$
  $t_{prev} \leftarrow t_c$

Note that Algorithm 2 ensures only new information is transmitted to the robots in $\mathcal{C}_i(t)$ without making unnecessary communications. Using similar reasoning to Lemma 4.2.2, Lemma 4.5.1 demonstrates that Algorithm 2 converges to a locally optimal solution to (4.2.4).

Lemma 4.5.1. *A change in goal locations between any pair of robots i and j in Algorithm 2 results in a decrease of the sum of distance remaining squared.*

Proof. After two robots trade their assigned goals, it is necessarily the case that:

$$\mathbf{u}_{ij}{}^{\mathrm{T}}\mathbf{w}_{ij} > 0$$

Using algebra similar to that in Lemma 4.2.2:

$$||\mathbf{r}_{ii}||^2 + ||\mathbf{r}_{jj}||^2 < ||\mathbf{r}_{ij}||^2 + ||\mathbf{r}_{ji}||^2$$

When a pair of robots exchange goal locations, the sum of distance squared remaining for those two agents decreases. All other robots are unaffected by the trade so the total sum of distance remaining squared decreases for the whole system. □

Theorem 4.5.2. *Algorithm 2 navigates each robot to a goal without any collisions.*

Proof. Define the cost-to-go function V:

$$V = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{g}_i||^2 \tag{4.5.3}$$

For an arbitrarily small constant, η, use Lemma 4.5.1 and the trajectories defined in (4.5.2) to see that V is strictly decreasing:

$$V(t+\eta) < V(t)$$

Further, by (4.5.2), the final value of the cost-to-go function will be zero:

$$V(t_f) = 0.$$

From Theorem 4.2.3, all trajectories will be free of collisions if all assumptions are met for every pairwise interaction. □

This pairwise algorithm is valid when the distance constraints in **(A4)** are met at the start of a pairwise interaction. However, there are pathological inputs that break these initial conditions. Section 4.8 addresses this in greater detail and provides two potential solutions. Note here that these conditions are extremely unlikely to arise in practice.

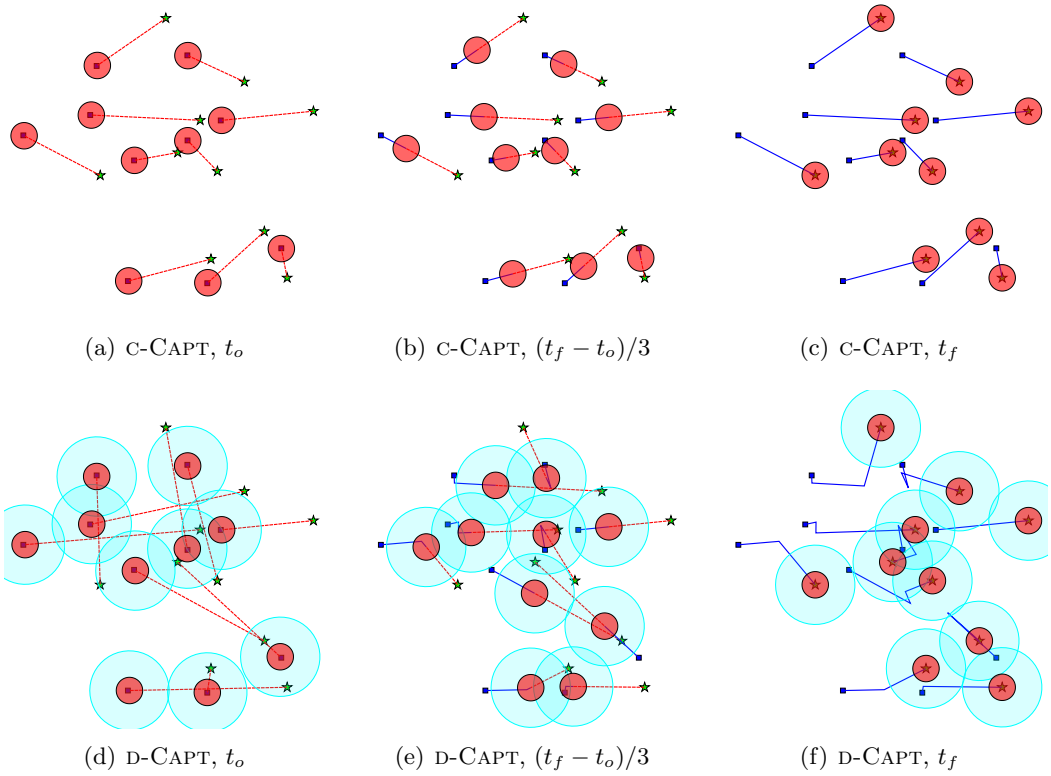
4.6 Simulation Results

This section presents a series of simulations of C-CAPT and D-CAPT on a large variety of boundary conditions to study their performance. Prior to each trial a series of randomly generated starting and goal locations are generated that satisfy **(A4)**.

The first set of simulations is shown in Fig. 4.5 comparing C-CAPT and D-CAPT on the same set of initial locations for 10 robots and 10 goal locations. In this example, $h = 1.1\Delta$, very near the lower limit ($h > \Delta$). In general, h should be much larger than this in an experimental system to account for real world uncertainties.

Shown in Fig. 4.6 is the verification that the energy function defined in (4.5.3) is indeed strictly decrescent by using a D-CAPT simulation with $N = M = 100$ in three dimensions with $h = 1.5\Delta$. It can be easily seen that $\dot{V} < 0$ and $V(t_f) = 0$.

The third set of simulation runs explore the scaling of D-CAPT as the number of robots increases. In Fig. 4.7 presents a number of trends with $\frac{h}{\Delta} >> 1$ in three dimensions. Note that the number of reassignments increases approximately linearly in N. The total number of communications grows approximately as N^2. Also computed is a comparison of the sum of distance traveled squared using D-CAPT to the optimal value returned from C-CAPT.



(a) C-CAPT, t_o (b) C-CAPT, $(t_f - t_o)/3$ (c) C-CAPT, t_f

(d) D-CAPT, t_o (e) D-CAPT, $(t_f - t_o)/3$ (f) D-CAPT, t_f

Figure 4.5: Comparing C-CAPT and D-CAPT simulations of 10 robots seeking 10 goal locations. In all figures, blue squares represent starting locations, green stars represent goal locations, small red circles represent the robots, red dotted lines indicate the planned trajectory, blue solid lines indicate the path already traversed, and cyan larger circles represent the communication range.

It should be noted that even with a fully connected network, optimality is not guaranteed with more than two robots. A simple example with three robots is presented in Fig. 4.8.

The next set of simulations in Fig. 4.9 vary communications ranges used by D-CAPT for $N = M = 20$. The communications range is varied from the minimum value of $\frac{h}{\Delta} = 1$ through large values ($\frac{h}{\Delta} >> 1$). Note that the number of messages sent decreases as the communications range decreases at the cost of becoming quite suboptimal. Additionally, the minimum clearance ϵ between robots decreases with smaller communications ranges

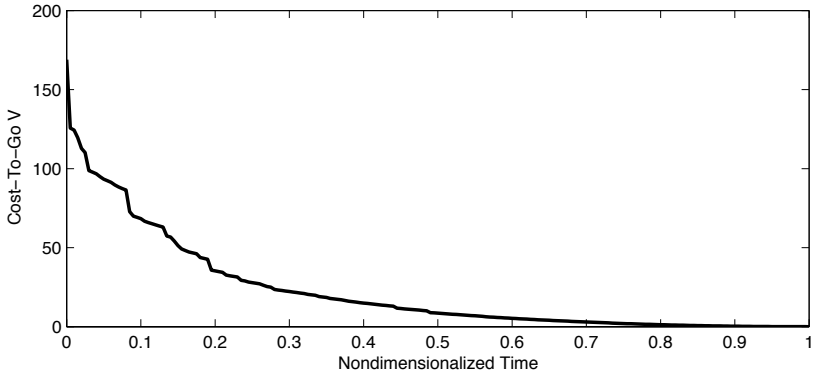


Figure 4.6: Visualization of decay of the cost-to-go function V in (4.5.3) for D-CAPT with $N = M = 100$, and $n = 3$. The instantaneous drops are a result of reassignments and the continuous decay is a result of trajectory tracking.

as one might expect, but always satisfy the clearance requirement in (4.1.6).

Finally demonstrated in Fig. 4.10 is C-CAPT with fewer agents than goals as outlined in Sect. 4.3. In this example, a brute force computation of the minimum distance solution is computationally infeasible. Instead, this image displays a feasible collision-free solution generated using Algorithm 1.

4.7 Experimental Verification

This section presents representative results from experiments in which a team of eight robots was tasked to visit a sequence of randomly generated goal locations. The robots used for experimentation are shown in Fig. 4.11. A bank of motion capture cameras provides feedback as shown in Fig. 4.12. Figure 4.13 shows the sequence of waypoints visited. All experiments resulted in safe execution and completion of the task. Minimum clearance between robots for the entire experiment is shown in Fig. 4.14. It is clear from

this plot that all robots were never in or near a collision state with any other robot.

4.8 Pathological Case for D-CAPT and Resolution

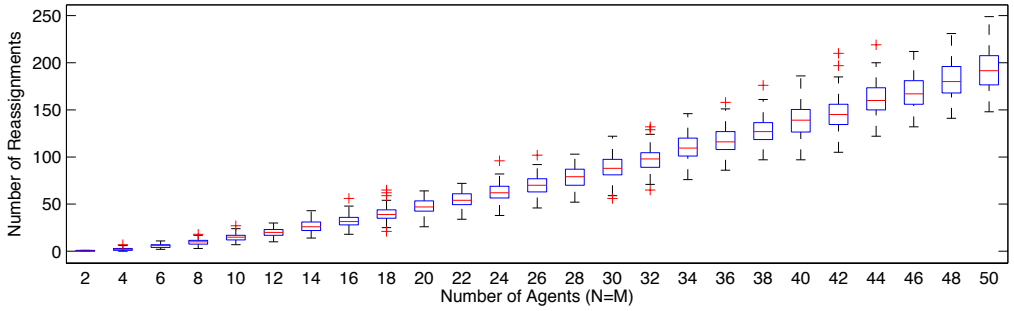
In D-CAPT presented in Sect 4.5, there is a class of initial conditions that can break the guarantee of collision avoidance for some pairs of robots. This pathological case arises when multiple unconnected networks of robots merge and the initial conditions in Eq. (4.1.3) are not satisfied. This happens only when at least one robots is out of communications range with the rest of the team while other networks are in very close proximity to one another and happen to break the initial condition requirement. This problem is addressed in more detail and a resolution is given in [68]. The most basic example of this case can be visualized in the three robot example in Fig. 4.15.

One potential solution is for all agents in the network to reverse directions long enough to ensure that the new trajectories are collision-free as displayed in Figs. 4.15(d)-4.15(f). This however requires all agents in the connected network to coordinate this reverse. Another solution is to control affected robots to spread out enough while remaining in $\mathcal{K}$ until Eq. (4.1.3) is satisfied and then resuming Algorithm 2 until convergence. A final approach is to use the recent results in [68].

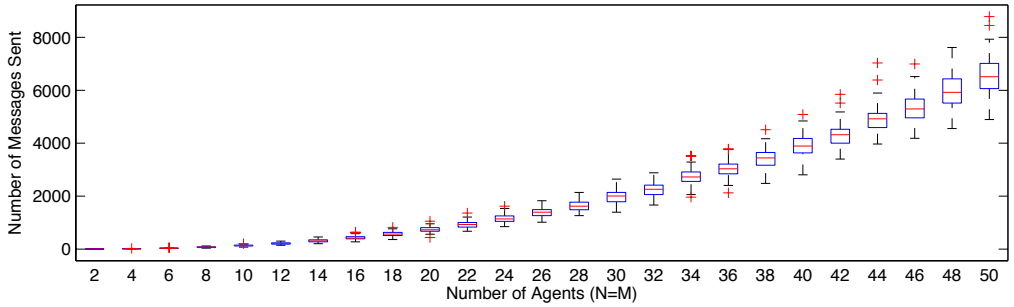
Fortunately, the conditions for this scenario to arise are extremely rare. Over hundreds of millions of simulated robot-robot communications using random starting locations, goals, and initial assignments, this scenario has never presented itself.

4.9 Chapter Summary

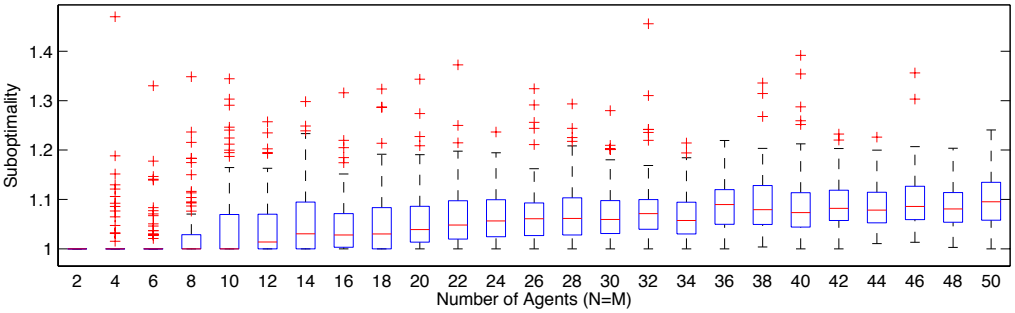
This chapter discusses the problem of Concurrent Assignment and Planning Trajectories, or CAPT, for multi-robot systems. First developed is C-CAPT, a centralized solution to the problem of assigning goals and planning trajectories that minimize a cost functional based on the square of velocity along the trajectory. The resulting trajectories are shown to be globally optimal and safe. C-CAPT is modified to be suitable for use on highly dynamic quadrotor micro aerial vehicle robots. Then, decentralized algorithm D-CAPT is derived and relies solely on the exchange of information between robots within communication range. This decentralized algorithm requires local reassignment and re-planning across a pair of robots when maximum distance traveled can be reduced while simultaneously increasing minimum clearance. Both algorithms are evaluated in simulation and C-CAPT is experimentally validated on a team of 8 quadrotor robots.



(a)

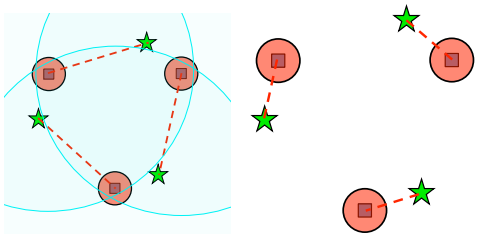


(b)



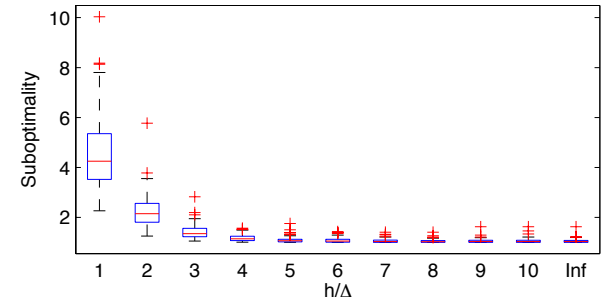
(c)

Figure 4.7: D-CAPT trends on simulated 3-dimensional random configurations and initial assignments for $M = N$ robots and goals. Each data point was tested with 100 random trials. Figure 4.7(c) shows the distance traveled squared divided by the optimal value returned from C-CAPT. The "+" marks designate statistical outliers.

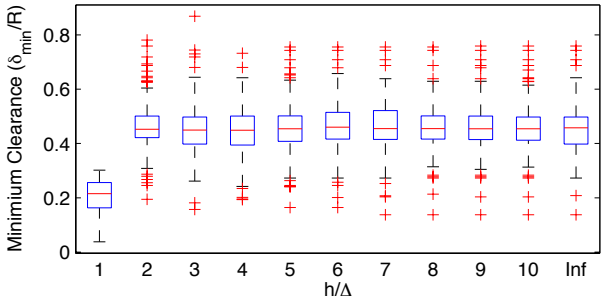


(a) (b)

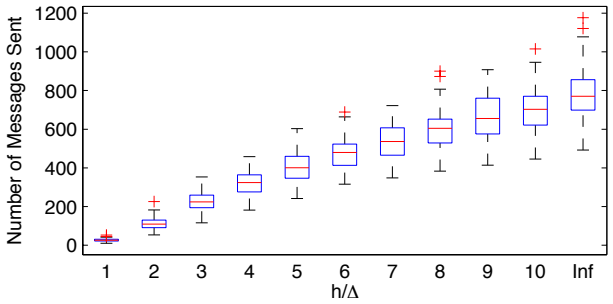
Figure 4.8: Simple case when D-CAPT does not find optimal solution for a fully connected team of $n > 2$ robots. For each pair of robots the scenario in Fig. 4.8(a), exchanging goals will not reduce the sum of distance squared. However, a permutation to modify all assignments to that in Fig. 4.8(b) does lower the sum of distance squared.



(a)

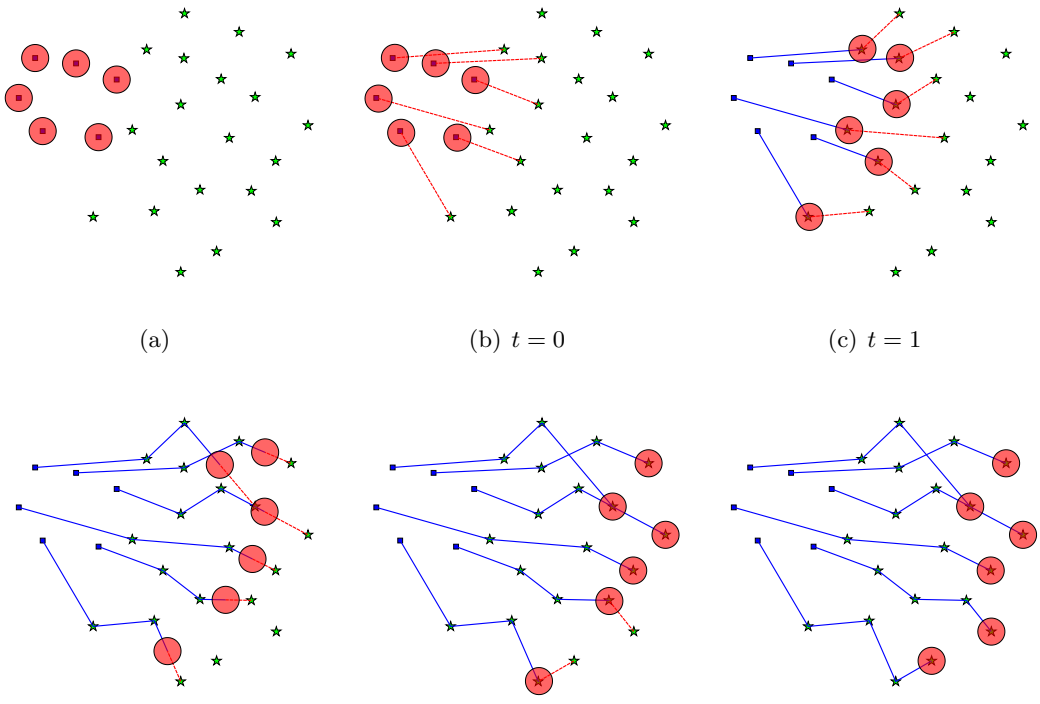


(b)



(c)

Figure 4.9: Box plots of D-CAPT statistics on 100 simulated 3-dimensional random configurations and initial assignments with varying communications distance h with N=M=20. Figure 4.9(c) shows that as the communications range decreases, the number of messages sent using D-CAPT drastically decreases at the expense of clearance in Fig. 4.9(b) and optimality in 4.9(a). Note that despite these small communications distances, the clearance requirement in (4.1.6) is never violated. "+" marks designate statistical outliers.



(a) (b) $t = 0$ (c) $t = 1$

(d) $t = 2.5$ (e) $t = 3$ (f) $t = 4$

Figure 4.10: c-Capt iteratively evaluating all goal locations when there are more goals than robots. Figure 4.10(a) shows the initial conditions with 6 robots and 20 goal locations. Figure 4.10(b) shows the first assignment of robots to goals and once this is completed, Fig. 4.10(c) shows another optimal assignment. Figure 4.10(d) displays the midpoint of the third segment for all robots. Figure 4.10(e) shows the case where there are now more remaining unvisited goals than robots and only some of the robots have assigned goals. Figure 4.10(f) shows the final state of system.

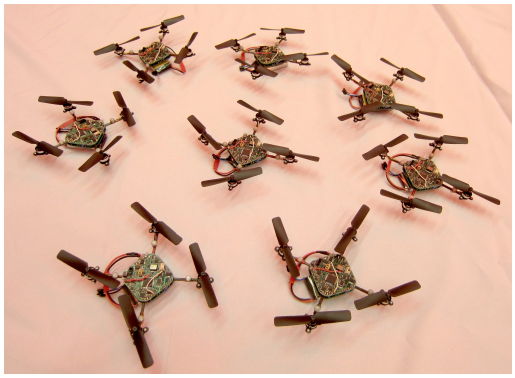


Figure 4.11: Team of eight KMel Robotics NanoQuad quadrotors used in experimentation. More information about these robots is available at www.kmelrobotics.com.

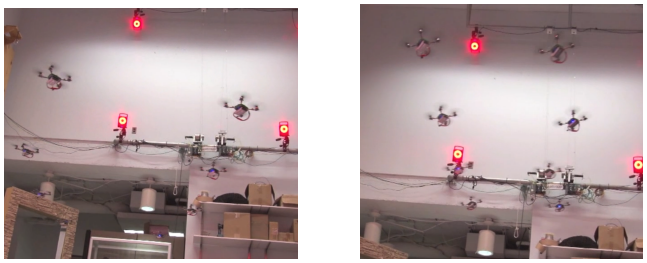


Figure 4.12: Eight robots flying in an indoor laboratory with feedback from motion capture cameras.

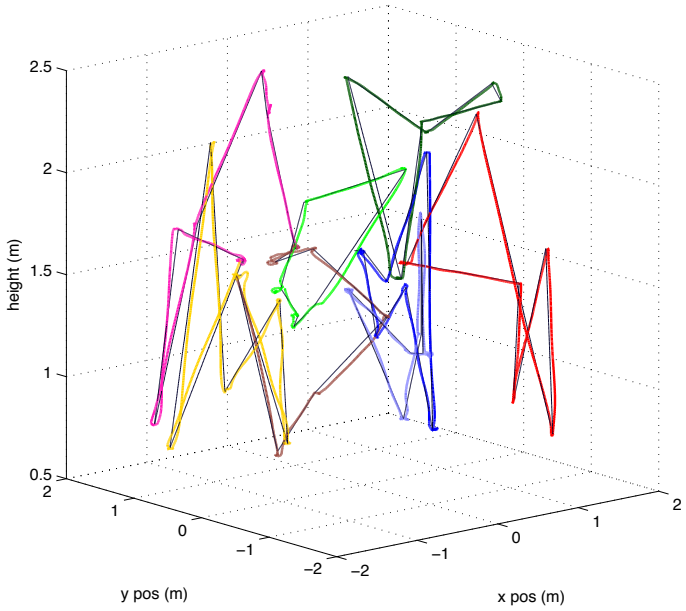


Figure 4.13: The 3 dimensional trajectories of the team of 8 quadrotor robots are shown here. The thin black lines show the desired trajectories and each colored line shows the actual paths followed of a particular robot.

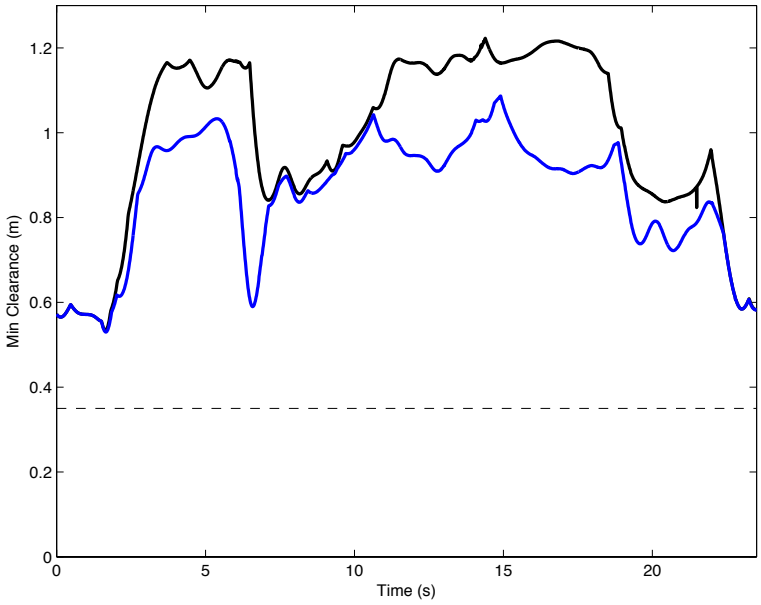


Figure 4.14: Plot of the minimum clearance between any two robots while tracking their trajectories. The upper black line represents the actual separation. The lower blue line represents a conservative parabolic estimate of the quadrotor shape, further explained in Appendix A. The dotted black line shows the actual diameter of the robots. Notice that collision avoidance is clear from the fact that the clearance is always larger than a robot's diameter.

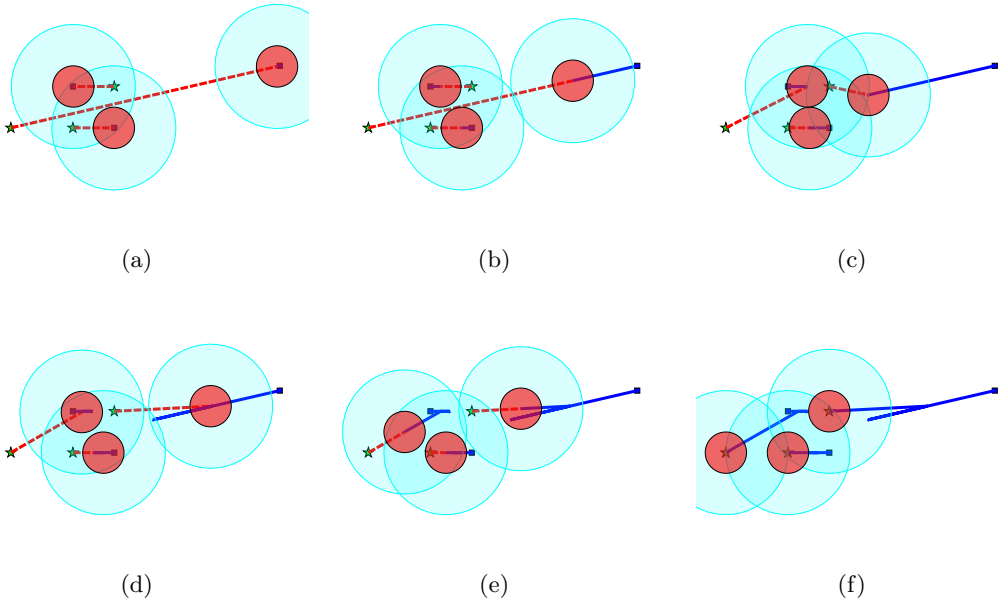


Figure 4.15: Figure 4.15(a) displays initial conditions which will result in a failure of D-CAPT. Figure 4.15(b) shows progress along the path, but Fig. 4.15(c) shows the time step immediately after the robots have converged to a connected graph and the left most robots both instantaneously make progress toward their goals. Figure 4.15(d) shows the robots reversing along their paths until Fig. 4.15(e) and 4.15(f) show the algorithm successfully navigating all robots to goals.

Chapter 5: Goal Assignment and Planning GAP

This chapter presents Goal Assignment and Planning (GAP): a computationally tractable, complete algorithm that generates dynamically feasible trajectories for N interchangeable robots navigating through known cluttered environments to M goal states. The computational complexity of this algorithm grows polynomially in the number of robots and goals- superior to the expected exponential complexity for multi-robot planning as discussed in Sect. 2.2. As a result, the GAP algorithm can be used to plan safe trajectories for dozens of highly dynamic robots in a few seconds.

Similar to C-CAPT from Chapter 4, GAP addresses the problem of finding both assignments and trajectories. However, working in obstacle filled environments requires a substantially different solution to this problem. The algorithm must find safe, time-parameterized trajectories which do not collide with obstacles or other robots. The first step in GAP is to plan for each robot individually as through no other robots existed. Therefore, this planning problem is equivalent to planning for a single robot. On first glance, this appears similar to the decoupled approaches discussed in Sect. 2.2. Although in the GAP algorithm, the single agent planner attempts to plan valid trajectories from a robot's initial state to *all* goal locations. Then, goals are assigned to minimize the maximum cost trajectory for any robot. Recall the intuition for minimizing the maximum cost from Sect. 2.5. Finally, the trajectories are given a final time parameterization which ensures collision avoidance between robots. It is important to note that the GAP algo-

rithm preserves the complete guarantees of the single robot planner (see Sect. 2.1 for a discussion on completeness) by leveraging the interchangeability of robots.

Minimizing the maximum cost is a particularly well suited method for tasks that must be performed as quickly as possible. For instance, imagine a search-and-rescue situation in which a number of robots must visit every room in a building to locate humans in need of assistance. Similarly, it is conducive for energy constrained robots to load share using min-max distance trajectories, even if the mission completion time is not minimized.

This chapter develops and extends the core ideas from Chapter 4 to incorporate planning in environments with static obstacles and for more complex dynamic systems. First, Sect. 5.1 introduces notation and explicitly lists all assumptions. The GAP algorithm is presented in Sect. 5.2. Section 5.3 presents a number of case studies to plan min-max distance safe trajectories for teams of two-dimensional kinematic robots. Then in Sect. 5.4, GAP is experimentally validated with a team of six quadrotor vehicles in a complex three-dimensional environment.

The work in this chapter originally appeared in [86, 88, 89].

5.1 Preliminaries

Let the set of natural numbers between 1 and Z be represented by $\mathcal{I}_Z \equiv \{1, 2, \ldots, Z\}$. Designate the state of robot $i \in \mathcal{I}_N$ as $\mathbf{x}_i \in \mathcal{X}$ where $\mathcal{X}$ is the state space of a single robot. While the state space often represents the physical location of a robot, it can also contain a number of other properties about the state of the robot including orientation, velocity, and angular rates. Define the initial state of robot i as $\mathbf{s}_i = \mathbf{x}_i(0)$. Similarly, $\mathbf{g}_j$ specifies the $j \in \mathcal{I}_M$ desired goal state.

The set of all points in Euclidean space occupied by robot i is represented by the open set $B(\mathbf{x}_i) \subset \mathbb{R}^d$, where d is the dimensionality of the robot extent. With slight abuse of notation, $B(\tau) \subset \mathbb{R}^d$ is the set of all points swept out by a robot tracking trajectory $\tau : t \to \mathcal{X}$.

It should be stressed at this point that the algorithm will only work when robots have the ability to remain at states $\mathbf{s}_i$ and $\mathbf{g}_j$ indefinitely. This requirement may preclude the use of fixed wing aerial vehicles that require forward velocity to remain in flight.

The assignment of robots is represented by the mapping $\phi : \mathcal{I}_N \to \mathcal{I}_M \cup 0$ where each goal can be assigned to a maximum of one robot such that:

$$\phi(i) = \begin{cases} j & \text{if robot } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases}$$

For notational simplicity, ϕ_i will be used equivalently to $\phi(i)$.

5.2 Algorithm

The Goal Assignment and Planning (GAP) algorithm presented in Algorithm 3 synthesizes assignment and trajectory generation methods to generate collision-free trajectories for a large number of robots. It is assumed that each vehicle is locally executing a trajectory tracking controller and that trajectories are dynamically feasible.

Section 5.2.1 details the generation of graph G, which is embedded in $\mathcal{X}$, the state space of a single robot. GAP then computes the cost of the optimal paths for each robot to every goal in the absence of other robots, a process discussed in Sect. 5.2.2. Section 5.2.3 defines the optimal assignment as that which solves the Lexicographic Bottleneck Assignment Problem. Next, Sect. 5.2.4 introduces the notion of prioritization of robots, which is

Algorithm 3 Goal Assignment and Planning $(G, \mathbf{s}, \mathbf{g})$

for $i \in \mathcal{I}_N$ **do**
  for $j \in \mathcal{I}_M$ **do**
    Compute $\gamma_{ij}(t)$, optimal trajectory from $\mathbf{s}_i$ to $\mathbf{g}_j$
Compute ϕ^*, the optimal assignment of goals to robots
for $i \in \mathcal{I}_N$ **do**
  for $j \in \mathcal{I}_N \setminus i$ **do**
    if $\mathbf{s}_i \in P^\star(s_j, g_{\phi_j^\star})$ **then**
      Assign partial order $\mathcal{P}$: $j \succ i$
    if $\mathbf{g}_i \in P^\star(s_j, g_{\phi_j^\star})$ **then**
      Assign partial order $\mathcal{P}$: $i \succ j$
Construct suitable total ordering ψ from partial ordering $\mathcal{P}$
Optional refinement of $\gamma_{i\phi_i^\star}(t)$. See Sect. 5.2.6
for $i = 1 \rightarrow N$ **do**
  Compute $\hat{t}_{\psi_i}$, time offset of robot with priority i
return trajectories $\gamma_{i\phi_i^\star}(t - \hat{t}_{\psi_i})$

always possible due to the minimum maximum cost property of the optimal assignment. In Sect. 5.2.5, trajectories are computed that satisfy robot dynamics and avoid collisions with the environment and other robots. Optional trajectory refinement is presented in Sect. 5.2.6. Finally, completeness of Algorithm 3 is shown in 5.2.7 and a complexity analysis is performed in Sect. 5.2.8.

5.2.1 Graph Generation

To facilitate single agent planning as discussed in Sect. 2.1, a directed graph $G = (V, E)$ is constructed by sampling the state space using either a deterministic or probabilistic planner such as a Probabilistic Roadmap (PRM). A planner which is resolution complete will find a solution if one exists if the resolution of the underlying graph is sufficiently fine. Similarly, probabilistic completeness signifies that the probability of a random sampling motion planner will fail to return a path if one exists approaches zero as the number of samples goes to infinity. Although this chapter only presents proof of resolution com-

pleteness, GAP also permits probabilistically complete planning.

Each vertex $v_i \in V$ represents a valid, collision-free state in $\mathcal{X}$ and V must contain all starting and goal states:

$$\mathbf{s}_i \in V \ \forall \ i \in \mathcal{I}_N, \qquad \mathbf{g}_j \in V \ \forall \ j \in \mathcal{I}_M \tag{5.2.1}$$

As discusse previously, an edge $e_{ij} \in E$ corresponds to a collision-free trajectory segment $\tau_{ij}(t)$ where $\tau_{ij}(t_0) = v_i$, $\tau_{ij}(t_f) = v_j$ and has cost $C(\tau_{ij}) > 0$. The underlying dynamics of the system will determine how to best compute these segments. Possible cost functions C include distance traveled, energy used, and trajectory duration.

The graph G cannot include edges e_{kl} corresponding to trajectories τ_{kl} that would result in a collision with a robot at states $\mathbf{s}_i$ or $\mathbf{g}_j$ unless one of the vertices v_k or v_l is $\mathbf{s}_i$ or $\mathbf{g}_j$:

$$\begin{aligned} v_k \neq \mathbf{s}_i \text{ AND } v_l \neq \mathbf{s}_i \implies B(\tau_{kl}) \cap B(\mathbf{s}_i) = \emptyset \\ v_k \neq g_i \text{ AND } v_l \neq g_i \implies B(\tau_{kl}) \cap B(g_i) = \emptyset \end{aligned} \tag{5.2.2}$$

For example, a regular orthogonally-connected grid for spherical robots satisfies Eq. (5.2.2).

A path in the graph G is represented by an ordered list of its edges, $P(v_i, v_j) = e_{ik}, e_{kl} \ldots, e_{mn}, e_{nj}$. The cost of a path in the graph is represented using the norm:

$$||P(v_i, v_j)|| = \sum_{e_{kl} \in P(v_i, v_j)} C(\tau_{kl})$$

The optimal path is that which minimizes the sum of costs of edges between vertices:

$$P^\star(v_i, v_j) = \underset{P(v_i, v_j)}{\arg\min} \quad ||P(v_i, v_j)||$$

The corresponding trajectory for a path from the graph can be constructed by the concatenation of trajectory segments. For example, $\gamma_{ij}(t)$ is the optimal trajectory from $\mathbf{s}_i$ to $\mathbf{g}_j$ and can be written as the sequence: $\gamma_{ij} = \{\tau_{ik}, \tau_{kl}, \ldots, \tau_{mn}, \tau_{nj}\}$.

Sufficient Conditions for Resolution Completeness To ensure resolution completeness, it is critical that a path generated by the single robot planning algorithm reflect the possibilities of the full muti-robot system. This section presents one set of sufficient conditions that ensure the resolution completeness of GAP.

First, define Y_i as the set of all valid states that, if occupied by a robot, would be in collision with a robot at $\mathbf{s}_i$. For a team of circular robots with radius R, Y_i would be the set of all vertices closer than $4R$ from s_i. Similarly, define W_i as the set of all valid states that, if occupied by a robot, would be in collision with a robot at $\mathbf{g}_i$. It will be shown that GAP is resolution complete if the following conditions are respected. First, every valid state in Y_i must be reachable from $\mathbf{s}_i$ without colliding with a robot at $\mathbf{s}_j$, for all $i \neq j$. A related constraint is that $\mathbf{s}_i$ must be reachable from every valid state in Y_i without colliding a robot at $\mathbf{s}_j$, for all $i \neq j$. Additionally, these constraints must also hold for the set of goal conditions.

Mathematically, these conditions can be represented by the following sets of equations:

$$\begin{aligned} &\forall v_j \in Y_i \ \exists \tau_{ij} \,\big|\, v_i = \mathbf{s}_i, B(\tau_{ij}) \cap B(\mathbf{s}_k) = \emptyset \ \forall \ k \in \mathcal{I}_N \setminus i \\ &\forall v_j \in Y_i \ \exists \tau_{ji} \,\big|\, v_i = \mathbf{s}_i, B(\tau_{ji}) \cap B(\mathbf{s}_k) = \emptyset \ \forall \ k \in \mathcal{I}_N \setminus i \end{aligned} \tag{5.2.3}$$

$$\begin{aligned} &\forall v_j \in W_i \ \exists \tau_{ij} \,\big|\, v_i = \mathbf{g}_i, B(\tau_{ij}) \cap B(\mathbf{g}_k) = \emptyset \ \forall \ k \in \mathcal{I}_M \setminus i \\ &\forall v_j \in W_i \ \exists \tau_{ji} \,\big|\, v_i = \mathbf{g}_i, B(\tau_{ji}) \cap B(\mathbf{g}_k) = \emptyset \ \forall \ k \in \mathcal{I}_M \setminus i \end{aligned} \tag{5.2.4}$$

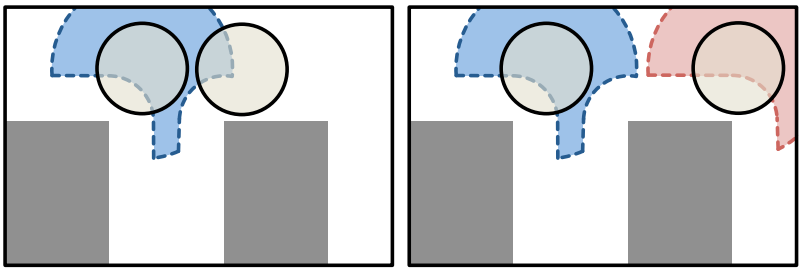
An intuition for these conditions is that a robot at its start or goal state should not modify

the reachable configuration space of other robots. For kinematic fully actuated robots, conditions (5.2.3) and (5.2.4) reduce to:

$$Y_i \cup Y_j \ \emptyset \ \forall i \neq j$$

$$W_i \cup W_j \ \emptyset \ \forall i \neq j$$

See Fig. 5.1 for an example of initial conditions which violate the assumption in Eq. (5.2.3) and a slightly modified environment which respects this condition.



(a) (b)

Figure 5.1: In each of Fig. 5.1(a) and Fig. 5.1(b), example initial locations for two-dimensional circular robots are shown. Tan circles represent the initial states of the two robots and the gray rectangular regions indicate obstacles. The dotted regions represent Y. Figure. 5.1(a) shows an example of poor initial conditions which do not satisfy Eq. (5.2.3). This is clear since to reach the rightmost side of Y for the robot on the left results in a collision with the robot on the left. Figure 5.1(b) slightly modifies the initial conditions such that there exists a valid trajectory from the starting location to every point in the region Y and therefore this initial condition is valid.

5.2.2 Computing Individual Trajectories

After construction of graph G, the graph is searched for the minimum cost path for all NM combinations of robots and goal states. Dijkstra's algorithm [15] is a natural choice for this graph search as it returns optimal paths in the graph from the start vertex of a robot to all other vertices.

After the optimal path through the graph is found, full robot trajectories can be

constructed from the trajectory segments τ_{kl} corresponding to $e_{kl} \in E$ to form $\gamma_{ij}(t)$ for robot i such that $\gamma_{ij}(t_{i,0}) = \mathbf{s}_i$ and $\gamma_{ij}(t_{i,f}) = \mathbf{g}_j$. Each robot will remain at its starting vertex when $t < t_{i,0}$ and its goal vertex if $t > t_{i,f}$.

If the graph search finds that goal $\mathbf{g}_j$ is unreachable from start location $\mathbf{s}_i$, the cost $C(\gamma_{ij})$ is infinite.

5.2.3 Assignment

The optimal assignment $\phi^\star$ is defined as that which solves the Lexicographic Bottleneck Assignment Problem (LexBAP) [8]. Similar to the standard bottleneck assignment problem, the LexBAP solution finds the feasible solution that minimizes the largest cost. Additionally, the LexBAP minimizes the second highest cost, as well as the third highest, and so on. Using this cost function ensures any pairwise exchange of assignment between robots i and j will not result in a decrease of maximum trajectory cost.

$$\max\left(||P^\star(\mathbf{s}_i, g_{\phi_i^\star})||, ||P^\star(s_j, g_{\phi_j^\star})||\right) \leq \max\left(||P^\star(\mathbf{s}_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})||\right) \tag{5.2.5}$$

The best solutions to the LexBAP have computational complexity of $\mathcal{O}(N^4)$ [78]. An intuition for the choice of the LexBAP is that it minimizes the L_∞ norm of the cost. This extends the concept in Chapter 4 to minimize the L_2 norm.

In practice, it is possible to relax the optimal LexBAP and use the solution of the linear assignment problem. This can be solved using the well-known Hungarian algorithm [41] with bounded computational complexity of $\mathcal{O}(N^3)$. The modified cost function minimizes the p-norm of the costs incurred by the team:

$$\phi^* \equiv \underset{\phi}{\operatorname{argmin}} \left(\sum_{i \in \mathcal{I}_N} ||P(\mathbf{s}_i, g_{\phi_i}||^p \right)^{\frac{1}{p}} \tag{5.2.6}$$

where p is sufficiently large such that the minimum-maximum cost principle in (5.2.5) holds. The solution to the modified linear assignment problem can be checked in $\mathcal{O}(N^2)$ time to determine if the assignment in (5.2.6) satisfies (5.2.5). Ideally, p is as large as possible, but unfortunately the size of p is bounded due to numerical issues. If (5.2.5) is not satisfied, the $\mathcal{O}(N^4)$ LexBAP solution can be utilized. In all of the simulation and experimentation trials used, a value of $p = 50$ was sufficient to satisfy (5.2.5) and the LexBAP solver was never required.

Recall that $\phi_i = 0$ signifies that robot i in not assigned to visit any goal location. The cost of a path to $\mathbf{g}_0$ is defined to be a value larger than any feasible cost. This choice will force as many robots as possible to navigate to a goal state. Excess goals will not be assigned to any robot through this assignment procedure and Algorithm 3 must be used iteratively to ensure all goals are eventually visited.

5.2.4 Prioritization

The prioritization stage induces an ordering ($\prec$) among all of the robots:

$$\mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}) \quad \Longrightarrow \quad i \prec j \qquad \forall i \neq j \tag{5.2.7}$$

$$\mathbf{g}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}) \quad \Longrightarrow \quad i \succ j \qquad \forall i \neq j \tag{5.2.8}$$

Theorem 5.2.1. *The conditions in Eq.* (5.2.7) *and Eq.* (5.2.8) *induce partial ordering* $\mathcal{P}$.

Proof. This ordering inherently is reflexive and the properties of antisymmetry and transitivity are shown in Lemma 5.2.2 and Lemma 5.2.3, respectively. Therefore the conditions in Eq. (5.2.7) and Eq. (5.2.8) induce a valid partial ordering. □

Lemma 5.2.2. *Eq.* (5.2.7)-(5.2.8) *satisfy the antisymmetry property* $i \prec j \implies i \not\succ j$

Proof. Assume $i \prec j$ and $i \succ j$. Without loss of generality, this requires one of the following cases:

$$\begin{aligned}
&\textsc{Case 1:} \quad \mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{s}_j \in P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star}) \\
&\textsc{Case 2:} \quad \mathbf{g}_{\phi_j^\star} \in P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star}), \mathbf{g}_{\phi_i^\star} \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}) \\
&\textsc{Case 3:} \quad \mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{g}_{\phi_i^\star} \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star})
\end{aligned}$$

First, consider Case 1. This implies both of the following equalities:

$$\begin{aligned}
||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star})|| &= ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})|| \\
||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star})|| &= ||P^\star(\mathbf{s}_i, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})||
\end{aligned}$$

Applying these to Eq. (5.2.5) yields:

$$\begin{aligned}
&\max\Big(||P^\star(\mathbf{s}_i, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})||, ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||\Big) \\
&\leq \max\Big(||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||, ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})||\Big)
\end{aligned}$$

This is clearly a contradiction and therefore Case 1 is not possible.

It can be similarly shown that Case 2 is also impossible using a change of variables.

Next, consider Case 3. Due to the optimal graph search, either

$$P(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star}) \subset P(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}) \quad \text{OR} \quad P(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}) = P(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star}) + P(\mathbf{g}_{\phi_i^\star}, \mathbf{s}_i) + P(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})$$

Regardless of which of these conditions is present, it is relatively straightforward to show that Eq. (5.2.5) is violated in much the same manner as Case 1 was handled.

As there are no conditions which allow for both $i \prec j$ and $i \succ j$, it is clear that $i \prec j \implies i \not\succ j$ □

Lemma 5.2.3. *The conditions in* (5.2.7)-(5.2.8) *satisfy the transitivity property* $i \prec j$ *AND* $j \prec k \implies i \prec k$

Proof. Assume $i \prec j$ AND $j \prec k$ AND $k \prec i$

Without loss of generality, there are the following conditions which result in this:

$$\begin{aligned}
&\textsc{Case 1:} \quad \mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{s}_j \in P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star}), \mathbf{s}_k \in P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star}) \\
&\textsc{Case 2:} \quad \mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{s}_j \in P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star}), \mathbf{g}_i \in P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star}) \\
&\textsc{Case 3:} \quad \mathbf{s}_i \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{g}_k \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{g}_i \in P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star}) \\
&\textsc{Case 4:} \quad \mathbf{g}_j \in P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star}), \mathbf{g}_k \in P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star}), \mathbf{g}_i \in P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star})
\end{aligned}$$

First, consider Case 1. The following conditions result:

$$\begin{aligned}
||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star})|| &= ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})|| \\
||P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star})|| &= ||P^\star(\mathbf{s}_k, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_k^\star})|| \\
||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star})|| &= ||P^\star(\mathbf{s}_i, \mathbf{s}_k)|| + ||P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_i^\star})||
\end{aligned} \tag{5.2.9}$$

it can be shown that the equalities (5.2.9) cannot be a solution to the LexBAP and:

$$\left(\sum_{l \in \{i,j,k\}} ||P(\mathbf{s}_l, \mathbf{g}_{\phi_l^\star})||^p \right)^{\frac{1}{p}} > \left(||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||^p + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_k^\star})||^p + ||P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_i^\star})||^p \right)^{\frac{1}{p}}$$

Therefore, Case 1 is not a valid set of conditions.

Next, consider Case 2:

$$\begin{aligned}
||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star})|| &= ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})|| \\
||P^\star(\mathbf{s}_k, \mathbf{g}_{\phi_k^\star})|| &= ||P^\star(\mathbf{s}_k, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})|| + ||P^\star(\mathbf{g}_{\phi_i^\star}, \mathbf{g}_{\phi_k^\star})||
\end{aligned} \tag{5.2.10}$$

Applying Eq. (5.2.5) for robots k and j while utilizing Eq. (5.2.10) yields:

$$\begin{aligned}
&\max(||P^\star(\mathbf{s}_k, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})|| + ||P^\star(\mathbf{g}_{\phi_i^\star}, \mathbf{g}_{\phi_k^\star})||, ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||) \\
&\leq \max(||P^\star(\mathbf{s}_k, \mathbf{s}_j)|| + ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||, ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})|| + ||P^\star(\mathbf{g}_{\phi_i^\star}, \mathbf{g}_{\phi_k^\star})||)
\end{aligned} \tag{5.2.11}$$

Make the following definitions:

$$\begin{aligned}
a &= ||P^\star(\mathbf{s}_k, \mathbf{s}_j)|| \quad b = ||P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_i^\star})|| \quad c = ||P^\star(\mathbf{g}_{\phi_i^\star}, \mathbf{g}_{\phi_k^\star})|| \\
d &= ||P^\star(\mathbf{s}_j, \mathbf{s}_i)|| \quad e = ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star})|| \quad f = ||P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_j^\star})||
\end{aligned} \tag{5.2.12}$$

Then, rewrite Eq. (5.2.11) using these definitions:

$$\max\left((a+b+c),(d+f)\right) \leq \max\left((a+d+f),(b+c)\right)$$

If $b+c \geq a+d+f$, it is clear from the previous equation that $b+c \geq a+b+c$, which implies $c \leq 0$. As c is defined as $||P^{\star}(\mathbf{g}_{\phi_i^{\star}}, \mathbf{g}_{\phi_k^{\star}})||$, this value must be strictly greater than zero. Therefore, the fact that $a+d+f \geq b+c$ is used:

$$\max\left((a+b+c),(d+f)\right) \leq a+d+f$$

Clearly $a+d+f \geq d+f$, so this can be simplified to:

$$a+b+c \leq a+d+f$$

and further to:

$$d+f \geq b+c \tag{5.2.13}$$

Applying Eq. (5.2.5) for robots i and j and using Eq. (5.2.10):

$$\max\left(||P^{\star}(\mathbf{s}_i, \mathbf{g}_{\phi_i^{\star}})||, ||P^{\star}(\mathbf{s}_j, \mathbf{s}_i)|| + ||P^{\star}(\mathbf{s}_i, \mathbf{g}_{\phi_j^{\star}})||\right) \leq \max\left(||P^{\star}(\mathbf{s}_i, \mathbf{g}_{\phi_j^{\star}})||, ||P^{\star}(\mathbf{s}_j, \mathbf{g}_{\phi_i^{\star}})||\right)$$

Using the definitions in (5.2.12), this can be rewritten as:

$$\max\left(e,(d+f)\right) \leq \max\left(b,f\right)$$

It can be shown through simple contradiction that $b \geq f$ to simplify this equation to:

$$\max\left(e,(d+f)\right) \leq b$$

Which means that $b \geq d+f$. Incorporating the knowledge from (5.2.13), this implies that:

$$b \geq d+f \geq b+c \implies 0 \geq c$$

However, c is defined to be strictly greater than zero and therefore it is impossible for Eq. (5.2.4) to be satisfied and therefore CASE 2 is impossible.

CASE 3 is equivalent to CASE 2 with a change of variables and is not possible.

CASE 4 is equivalent to CASE 1 with a change of variables and is not possible.

As there are no conditions which allow $i \prec j$ AND $j \prec k$ AND $k \prec i$, then it is clear that $i \prec j$ AND $j \prec k \implies i \prec k$. Therefore the conditions in Eq. (5.2.7) and Eq. (5.2.8) satisfy the transitivity property. □

The next step is to construct a total ordering which respects all relations of the partial ordering $\mathcal{P}$. In the case of an ambiguity in the partial ordering, the cost of the trajectories can be used as a tie breaker when constructing this total ordering. This total ordering can then be represented by the mapping $\psi : \mathcal{I}_N \to \mathcal{I}_N$ where ψ_i identifies the robot with priority i.

5.2.5 Parameterization and Collision Avoidance

At this point, feasible trajectories have been computed for every robot-goal pair (γ_{ij}) with robots either assigned to a goal destination $\phi_i^\star = j \in \mathcal{I}_M$ or stationary $\phi_i^\star = 0$, and a prioritization order (ψ_i). The trajectories $\gamma_{i\phi_i^\star}(t)$ will navigate the robot from the initial state $\mathbf{s}_i$ to the final assigned state $\mathbf{g}_{\phi_i^\star}$. However at this point, there may be collisions between robots. This section details a reparametrization of the planned trajectories, making use of the resting boundary conditions assumption.

Similar to greedy direct sequential routing in [75], robots are systematically assigned time offsets, $\hat{t} \geq 0$, to avoid collision with robots of higher priority. The robot with highest priority (ψ_1) begins with an offset of $\hat{t}_{\psi_1} = 0$. Then, the robot with the second highest

priority (ψ_2) computes an offset such that its trajectory never collides with the robot with highest priority. It is usually best to minimize the offset times:

$$\hat{t}_{\psi_2} = \arg\min_{\hat{t}} \; B(\gamma_{\psi_2,\phi_{\psi_2}}(t-\hat{t})) \cap B(\gamma_{\psi_1,\phi_{\psi_1}}(t-\hat{t}_{\psi_1})) = \emptyset \tag{5.2.14}$$

Similarly, the robot with the third highest priority then finds the minimum time offset which results in a collision-free trajectory between it and all other robots with higher priority. This continues with each robot requiring knowledge of the full trajectory of all robots with higher priority than itself until all offset times are computed, $\hat{t}_i \; \forall \; i \in \mathcal{I}_N$. Lemma 5.2.4 demonstrates that such a time offset always exists to avoid collision with all robots.

Lemma 5.2.4. *Time offset $\hat{t}_i$ exists for all robots such that no collisions occur for robots to navigate to their assigned goal location.*

Proof. The robot with highest priority will be able to have a time offset of $\hat{t}_{\psi_1} = 0$. This robot will be able to navigate to its goal location using the trajectory corresponding to its assignment without colliding with any robots at their starting locations. If a collision were going to occur between robot ψ_1 and robot j remaining at its starting state, using Eq. (5.2.2) and Eq. (5.2.7) dictates that robot j would have higher priority and contradicting the fact that robot ψ_1 has highest priority.

Robot ψ_2 can find time offset $\hat{t}_{\psi_2} \geq 0$. Using a similar logic to that of robot ψ_1, robot ψ_2 will not collide with any robots having lower priority if they are to remain at their initial state. If $\hat{t}_{\psi_2}$ is large enough, robot ψ_1 will have arrived at its goal location. In this case, either robot ψ_2 cannot collide with robot ψ_1 or the condition in Eq. (5.2.2) results in a contradiction with Eq. (5.2.8).

Existence of time offsets for all other robots can be found by extension. □

In the worst case, Lemma 5.2.4 would have only one robot moving at a time, in order of priority. In practice, trajectories will complete much more quickly than this bound.

5.2.6 Optional Trajectory Refinement

While the algorithm presented thus far in Sect. 5.2 is very general for dynamic systems, systems with complex dynamics may require trajectory planning in the complete state space which may be very difficult, time consuming, or memory intensive to perform. Therefore, this section presents an optional method to refine trajectories, allowing a lower resolution graph to be constructed or even the use of a lower dimensional robot state.

An assigned trajectory $\gamma_{i\phi_i^\star}$ can be optionally refined to $\tilde{\gamma}_{i\phi_i^\star}$ immediately preceding calculation of time parameterization in Sect. 5.2.5.

There are four requirements for a valid trajectory refinement. First, the refined trajectories must be dynamically feasible and not cause a collision with any obstacle. Next, refined trajectories must satisfy the boundary conditions $\mathbf{s}_i$ and $\mathbf{g}_i$. Third, refined trajectories must respect the constraints specified by the computed total ordering ψ:

$$\psi_i > \psi_j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(\mathbf{g}_j) = \emptyset$$

$$\psi_i < \psi_j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(\mathbf{s}_j) = \emptyset$$

Finally, the refinement must decrease the cost of the trajectory for an individual robot:

$$C(\tilde{\gamma}_{i\phi_i^\star}) < C(\gamma_{i\phi_i^\star}) = \sum_{e_{ij} \in P^\star(\mathbf{s}_i, \mathbf{g}_{\phi_i^\star})} C(\tau_{ij})$$

If these conditions are met, the results from Theorem 5.2.5 continue to apply and collision avoidance is guaranteed. The basic steps of trajectory refinement are shown in Fig. 5.2.

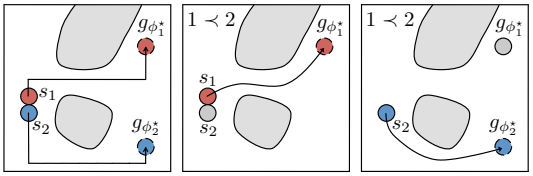


Figure 5.2: Optional refinement of trajectories. Figure 5.2(a) shows optimal initial trajectories and Fig. 5.2(b) shows trajectory refinement of a higher priority robot. The robot must avoid collisions with the starting states of all robots with lower priority. Figure 5.2(c) shows trajectory refinement of a lower priority robot. This robot must avoid goal states of higher priority robots.

5.2.7 Completeness

Theorem 5.2.5. GAP*, as presented in Algorithm 3, is resolution complete and returns collision-free trajectories tasking as many robots to goal states as possible.*

Proof. Resolution completeness requires finding a solution if one exists and otherwise correctly returning that there is no solution.

By design, all trajectories are dynamically feasible and do not result in robot-obstacle collision. The assumptions in Eq. (5.2.3) and (5.2.4) specify a robot at either a start or goal state cannot change the topology of the configuration space of any robot. Combining this with the fact that Dijkstra's algorithm is resolution complete implies that if there is a solution for robot i to reach goal $\mathbf{g}_j$ in the coupled planning problem, it will be found using the single robot planner in a sufficiently sampled graph. If there is not a trajectory for any robot to reach goal $\mathbf{g}_j$ in the coupled planning problem, clearly the single agent planner will not find it.

The optimal task assignment presented in Sect. 5.2.3 ensures that as many goals as possible will be visited. The partial ordering imposed in (5.2.7) and (5.2.8) guarantees

that a valid total ordering is always found for a given assignment. Finally, Lemma 5.2.4 states that a set of valid time offset exists for a valid total ordering.

Therefore, the GAP algorithm is resolution complete. □

5.2.8 Complexity Analysis

This section will analyze the computational complexity of the presented algorithm in terms of number of the robots, N, the number of goals, M, the number of vertices in the graph, $|V|$, and the number of edges in the graph, $|E|$. Sect. 5.2.2 outlines the systematic planning of individual robot trajectories and makes the assumption that G has already been constructed.

To search for the optimal solution using Dijkstra's algorithm using a min priority queue for each start location to all vertices has complexity bound of $\mathcal{O}(|E| + |V|\log|V|)$ [23]. Therefore, finding paths from all starting locations to all goal locations has bounded complexity of $\mathcal{O}(N(|E| + |V|\log|V|))$. As discussed in Sect. 5.2.3, LexBAP can be solved in $\mathcal{O}(\max(N, M)^4)$. Although, typically, the linear assignment problem solution will be used with a complexity bound of $\mathcal{O}(\max(N, M)^3)$. The prioritization scheme designed in Sect. 5.2.4 considers pairwise interactions between robots and therefore grows with a complexity bound of $\mathcal{O}(N^2)$. Computing time offsets in Sect. 5.2.5 grows with complexity bound $\mathcal{O}(N^2)$. The optional refinement step considers robots one at a time and has complexity bound $\mathcal{O}(N)$. Therefore, the worst case complexity of this algorithm is $\mathcal{O}(\max(N, M)^4 + N(|E| + |V|\log|V|))$. This computational complexity is quartic in the number of robots or goals, which will tend to dominate the solution time at high N or M. However, in practice, the bound tends to be dominated by the $\mathcal{O}(\max(N, M)^3)$ Hungar-

ian algorithm, which is the best known complexity bound for the optimal solution to the assignment problem. To visualize these bounds, Fig. 5.15 shows the computation times for a large number of simulated trials and the trends as the number of robots changes.

5.3 Case Study: Minimum Distance Paths for Two-Dimensional Robots

In this section, the problem of navigating a team of circular interchangeable two-dimensional robots is considered. Section 5.3.1 explores a simple example to show explicitly how each step of Algorithm 3 is computed. In Sect. 5.3.2, movement restrictions are lifted to show how the initial condition requirements affects the allowable graphs. For this system, a deterministic grid and a Probabilistic Road Map are compared. In Sect. 5.3.3, the unique minimum distance solution is considered as well as how to generate feasible trajectories for a robot with second order dynamics.

5.3.1 Explicit Example for a Team of Robots with Constrained Motion

This section considers a team of interchangeable circular kinematic robots with diameter 1 that are restricted to move along a unit length four-connected grid. The state is represented as a two-dimensional position vector, $\mathcal{X} = \mathbb{R}^2$ and the space occupied by each robot is a disk of radius 1/2. Additionally, all initial and goal locations for this system fall on a regular grid of edge length 1. The edges between neighboring cells have corresponding straight lines trajectories with constant velocity:

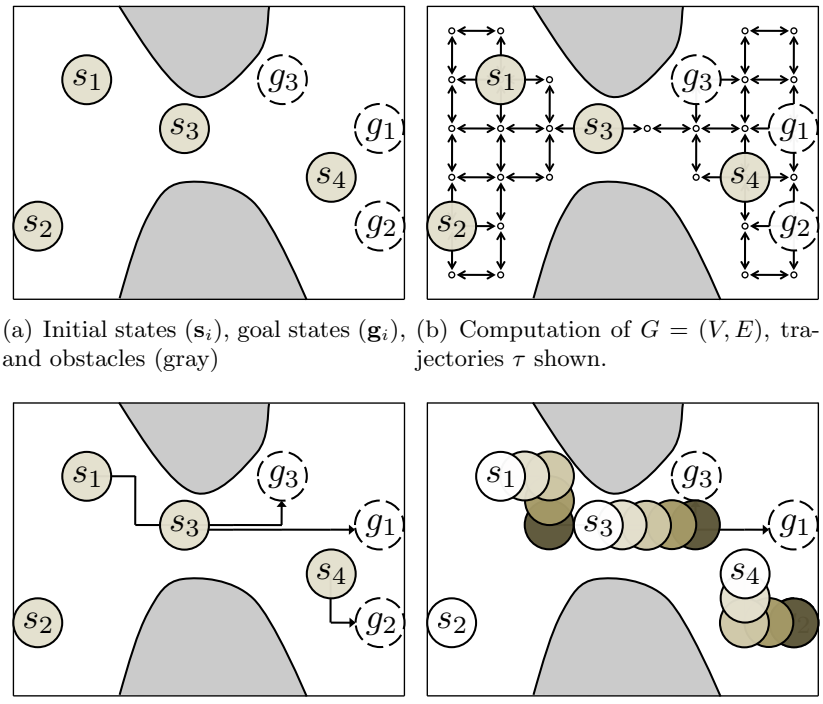
$$\tau_{ij}(t) = (v_i - v_j)\left(\frac{t - t_0}{t_f - t_0}\right)$$

For motivation for this system, see the Kiva Systems mobile fulfillment solution [19].

It is critical to ensure that this system respects the requirements specified in Sect. 5.1. Since all vertices fall along the grid, Eq. (5.2.1) is necessarily satisfied. The graph only contains orthogonal connections, resulting in each trajectory only causing collision with a vertex if that one of the endpoints of that trajectory are the vertex in the collision state. As this is satisfied for all vertices, it is satisfied for initial and goal vertices and satisfies Eq. (5.2.2). As all possible vertices and edges are included, Eq. (5.2.3) and Eq. (5.2.4) are not required for resolution completeness.

The edge costs are defined as the distance traveled:

$$C(\tau_{ij}) = ||v_i - v_j||_2 = D = 1$$



(a) Initial states ($\mathbf{s}_i$), goal states ($\mathbf{g}_i$), and obstacles (gray)

(b) Computation of $G = (V, E)$, trajectories τ shown.

(c) Optimal assignment trajectories ($\phi^\star$)

(d) Snapshots of trajectory tracking at $t = 0.5, 1.0, 1.5, 2.0$

Figure 5.3: Key algorithmic steps for a robotic team of $N = 4$ circular robots navigating to $M = 3$ goal states.

Now, consider the $N = 4$ robot system with $M = 3$ goals depicted in Fig. 5.3(a). The graph and corresponding trajectories are constructed making use of the regular grid as shown in Fig. 5.3(b). Then the optimal path through the graph is planned for each robot to every goal, resulting in 12 trajectories. The trajectory costs returned for each of the paths through the graph in Fig. 5.3(b) are:

$$\left[\ ||P^\star(\mathbf{s}_i, \mathbf{g}_j)||\ \right] = \begin{bmatrix} 7 & 9 & 6 \\ 9 & 11 & 8 \\ 4 & 6 & 3 \\ 2 & 2 & 3 \end{bmatrix}$$

The optimal assignment assigns robot 1 to goal 3, robot 3 to goal 1, and robot 4 to goal 2, or $\phi_1^\star = 3, \phi_2^\star = 0, \phi_3^\star = 1, \phi_4^\star = 2$. The trajectories correlating to these assignments are plotted in Fig. 5.3(c).

Notice how in Fig. 5.3(c), the starting location of robot 3 is in the path of robot 1, or $s_3 \in P(s_1, g_{\phi_1^\star})$. Therefore, according to (5.2.7), robot 3 has priority over robot 1, or a partial order is induced of $1 \succ 3$. A valid total ordering might be $\{3, 2, 1, 4\}$. In this case, $\psi_1 = 3$, $\psi_2 = 2$, $\psi_3 = 1$, and $\psi_4 = 4$. Now, robot ψ_1, or robot 3 assigns its time offset of $\hat{t}_3 = 0$. Then robot $\psi_2 = 2$ assignes its time offset, which can also be 0. Next, robot $\psi_3 = 1$ assigns its offset time, which can be $\hat{t}_1 = 0$ as well since the trajectory will not lead to a collision even with zero offset. Finally robot $\psi_4 = 4$ plans $\hat{t}_4 = 0$.

5.3.2 Constrained Movement Relaxation

At this point, we relax the assumption that all boundary locations fall on a unit-spaced grid and allow robots to move in any direction. A much greater variety of systems are now able to be considered compared to those in Sect. 5.3.1.

Next, consider the following conditions on the construction of the graph such that it satisfies Eq. (5.2.2), Eq. (5.2.3), and Eq. (5.2.4) for this fully actuated system.

First, the position of all starting locations are required to be at least $4R$ from any

other starting condition:

$$||\mathbf{s}_i - \mathbf{s}_j||_2 > 4R \quad \forall i \neq j \in \mathcal{I}_N$$

Similarly require all goal locations to be at least $4R$ from any other starting goal.

$$||\mathbf{g}_i - \mathbf{g}_j||_2 > 4R \quad \forall i \neq j \in \mathcal{I}_M$$

Denote the configuration space as $\mathcal{C}$ and consider the regions Y_i and W_i. These are the areas in free configuration space that are within $2R$ of the starting location i or goal location j as:

$$Y_i = \mathcal{C} \cap (\mathbf{s}_i \oplus \mathcal{B}_{2R})$$

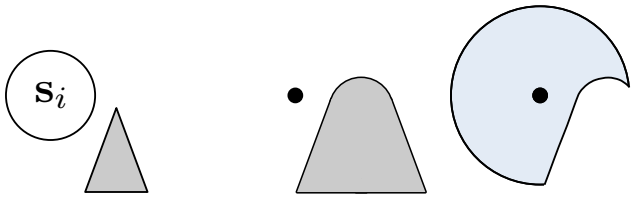
$$W_j = \mathcal{C} \cap (\mathbf{g}_j \oplus \mathcal{B}_{2R})$$

where $\oplus$ is the Minkowski sum. The initial locations are valid if the set Y_i is required to be a connected region around $\mathbf{s}_i$. If this is the case, there exists some continuous path from the initial positions to anywhere that is in collision with the initial location, including its boundary. Similarly, the set W_j $\forall j \in \mathcal{I}_M$ must be a connected region around point $\mathbf{g}_j$.

Enforcing the $4R$ condition ensures no other starting or goal states cause collision with any point in Y_i. By also ensuring Y_i $\forall i \in \mathcal{I}_N$ is a connected region, this means that the fully actuated robot can navigate from its initial state to any point in region Y_i. Therefore, this satisfies Eq. (5.2.3). Similarly, it can be shown that this condition satisfies Eq. (5.2.4).

As before, define the cost function for trajectory segments as distance between the end points and trajectories have constant velocity.

Figure 5.5 shows the graphs generated for an 8-connected grid as well as for a PRM with the same number of vertices and a similar number of edges. A prominent feature of



(a) Initial condition (b) Config. space, $\mathcal{C}$ (c) Region Y_i

Figure 5.4: Example of valid initial conditions. The initial location of robot i is specified as $\mathbf{s}_i$ in Fig. 5.4(a) and the gray triangle represents an obstacle. The configuration space $\mathcal{C}$ is the white region in Fig. 5.4(b). The region $Y_i = \mathcal{C} \cap (\mathbf{s}_i \oplus \mathcal{B}_{2R})$ is the blue region in Fig. 5.4(c). Every point in this region is reachable from $\mathbf{s}_i$ and is therefore valid.

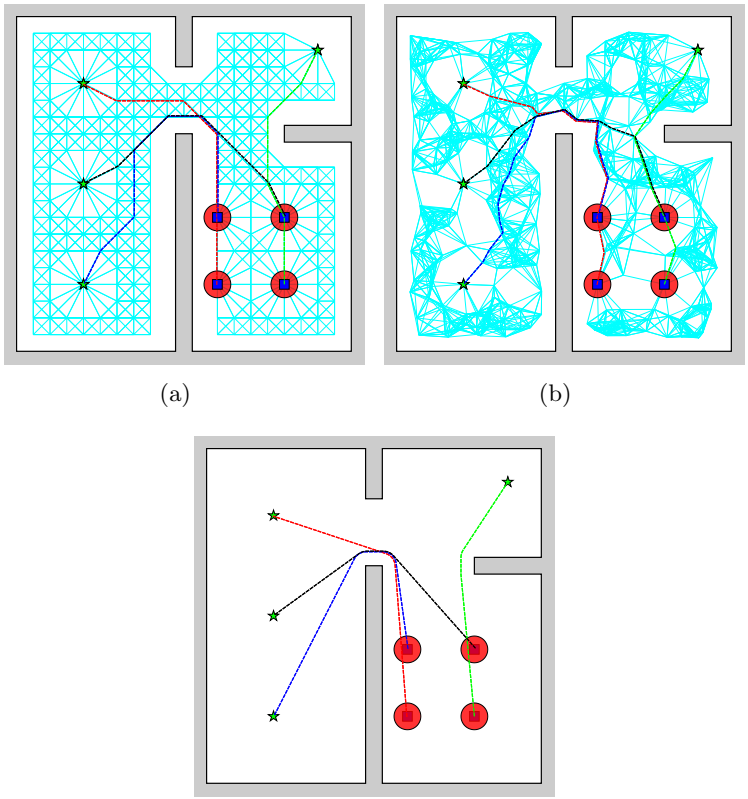
these two graphs is that no trajectories pass through the region within $2R$ of a start or goal location unless the trajectory has an endpoint which is the start or goal location.

Fig. 5.7 shows that for the environment which requires tight interaction between robots in Fig. 5.6 with 64 robots navigating on a PRM graph to 64 goals, clearance constraints are always satisfied ensuring trajectories are collision-free.

5.3.3 Optimal Solution With Second Order Dynamics

While experimental details are outside of the scope of this paper, [64] demonstrates that a team of interchangeable boats (See Fig. 5.8) with the capability to dock with other boats is shown to have great potential as quick response bridges or landing platforms. These boats utilize GAP and motivate the consideration of dynamics of the fully actuated second order systems.

It is well known that the unique minimum distance trajectory for an individual two-dimensional robot can be found using a visibility graph. The assumption that boundary conditions are no closer than $4R$ from each other satisfy the requirements in Eq. (5.2.2).



(a) (b)

(c)

Figure 5.5: Figure 5.5(a) shows a valid graph using an 8-connected grid with a grid-cell length of $D/2$ and the paths associated with the optimal assignment for $N = 4$ robots navigating to $M = 4$ goals. Figure 5.5(b) shows a PRM graph for the same environment. Figure 5.5(c) displays the unique minimum distance solution. Note that while it not guaranteed to be the case, all assignments are the same and the same partial orders are present using each of these graph generation methods.

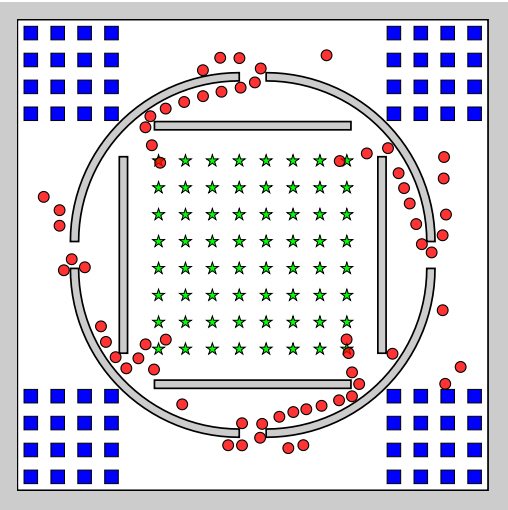


Figure 5.6: A team of 64 robots mid-simulation using a PRM in a tight environment, forcing a high degree of interaction.

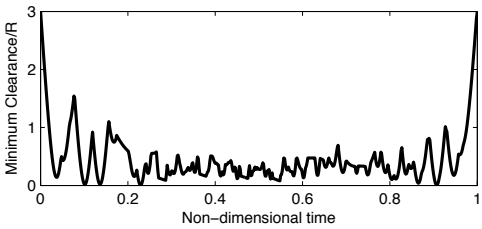


Figure 5.7: A plot of the minimum clearance between any two robots over the length of all trajectories for 64 robots in Fig. 5.6. Clearance is defined as the space between robots. Collision avoidance is evidenced by the fact that inter-robot and environment clearance is never negative.

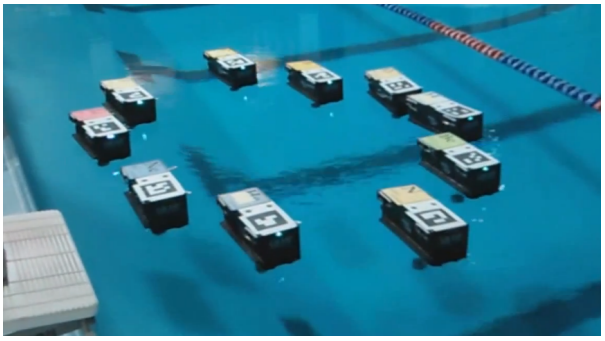


Figure 5.8: Video of aquatic robots available at `youtube.com/watch?v=2OY3nBtGqVU`, image courtesy of [64].

However, in order to ensure Eq. (5.2.3) and Eq. (5.2.4) are satisfied, the following conditions must be met. Not only must the regions Y_i and G_j be connected regions, but any trajectory passing within $2R$ of an initial or final condition must induce the ordering as if that trajectory went through the vertex. This is the same condition present for the work in [86], but can be shown to satisfy Eq. (5.2.3) and Eq. (5.2.4) after slight modifications and trajectory refinement. See Fig. 5.5(c) for an example of a unique minimum distance solution.

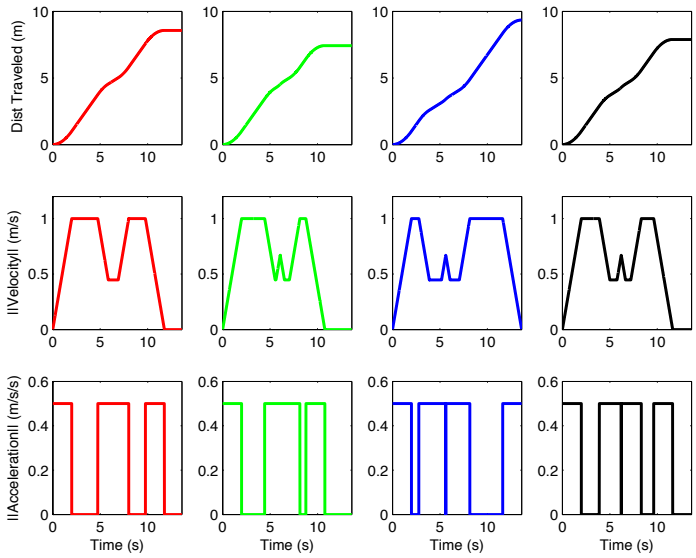


Figure 5.9: State of second order robots with maximum velocity of 1m/s and maximum acceleration of 0.5 m/s/s for the four robots in Fig. 5.5(c) with colors preserved. Note that the velocity slows around corners and that either velocity or acceleration is always at its maximum value.

For the unique minimum distance paths, $P^\star(\mathbf{s}_j, \mathbf{g}_{\phi_j^\star})$ will be $\mathbb{C}^1$ smooth as it will be a concatenation of straight lines and arcs. As such, it is straightforward to find the unique minimum distance paths and then parameterize them to allow robots with second order constrained dynamics to minimize the time to track that path. For instance, see Fig. 5.9

for time plots of position, velocity, and acceleration for a the team of robots in Fig. 5.5(c).

5.4 Experimental Results

The algorithm presented in this paper was implemented on a team of homogeneous quadrotors to navigate a complex maze-like environment. Figure 5.10 shows the 75 gram test vehicle used in these experiments. To guide the experiment design, consider a scenario where a team of quadrotors enters a structure, explores a series of chambers connected by narrow passageways, and departs the structure. The structure used for experimentation, shown in Fig. 5.11, has been designed to be reconfigurable with small passageways. For this experiment, the passageways are configured such that there exist very long internal paths within the structure.

Due to their dynamics, quadrotors require planning in a 12-dimensional state space to fully model the robots' capabilities. Fortunately, the quadrotor has been shown to be differentially flat with flat outputs of position and yaw: $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^3 \times SO(2)$. For the purpose of these experiments, orientation of the quadrotor is not relevant due to symmetry, so the state of the robots is the position vector $\mathcal{X} = \mathbb{R}^3$. Minimum snap trajectories have been shown to be very well suited to quadrotors [53, 61] and these trajectories can be generated through a number of waypoints quickly and reliably.

In a closed environment as in these experiments, aerodynamic effects such as ground and inter-vehicle effects are substantial for the quadrotors, particularly between multiple robots in a confined space. Therefore, $B(\mathbf{x})$ is represented by an ellipsoid with elongated vertical dimension to minimize downwash effects on other robots (as proposed in [56]) and further explored in Appendix A. However, as the robots are navigating in a closed space

with floors and ceilings in some of the chambers, additional considerations are required. The robots' extent is only valid within the chamber in which it is currently occupying.

The graph, G, is systematically generated by selecting a large number of vertices $v_i \in \mathbb{R}^3$ in and out of the structure which have zero velocity, acceleration, and jerk and respect the conditions (5.2.1), (5.2.2). While it would be theoretically possible to sample states in the full 12 dimensional state space, this is impractical in this setting. The trajectory τ_{ij} is generated by minimizing snap to navigate from one vertex to the next where the cost function used is the integral of snap over the trajectory:

$$C(\tau_{ij}) = \int_{t_o}^{t_f} ||\dddot{\mathbf{x}}_i(t)||^2 \, dt$$

Edge e_{ij} is added if $||v_i - v_j||^2$ is below a threshold distance and trajectory τ_{ij} is obstacle collision-free.

Then, optimal individual trajectories are computed by using Dijkstra's algorithm. Optimal assignment $\phi^\star$ and the total ordering ψ are then computed based on the optimal paths, boundary conditions, and path costs. Finally, time offsets are computed in order of priority ψ to ensure collision avoidance of trajectories. Additional time is added to each time offset robots to minimize the effects of lingering aerodynamic turbulence and downwash.

As the quadrotor has homogeneous boundary conditions at every vertex, the original trajectories tend to take a substantial amount of time. Therefore, trajectory refinement is used to complement the complex dynamics of the quadrotor. The zero velocity, acceleration, and jerk boundary conditions are relaxed where possible to allow the quadrotor to make faster progress and expend less energy. In many instances, this results in longer

distance traveled for each robot, but less energy expended and shorter mission completion times.

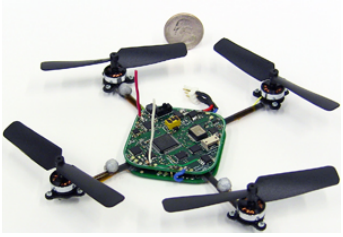


Figure 5.10: KMel Robotics NanoQuad quadrotor used in experimentation.

Figure 5.11: The structure used for experimentation with 12 chambers.

One experimental trial begins with 6 quadrotors hovering outside of the structure. Next, 6 goal states are specified, one goal state per chamber on the lower level of the structure. The trajectory for every robot is then computed using GAP (Algorithm 3). Figure 5.13 shows the trajectories computed, as well as the actual position of the robots in an experiment at one instant. Next, 6 additional goal states are given with one in each

chamber of the upper level and those trajectories are computed and then tracked. Finally, 6 goal states are specified on the outside of the structure. See Fig. 5.12 for an image of the 6 robots arriving at specified waypoints.

Video results of single and multiple robot experiments are available online[1]. This video also includes a simulation of 25 robots navigating a cluttered 3D environment, which was computed in 0.5 seconds.

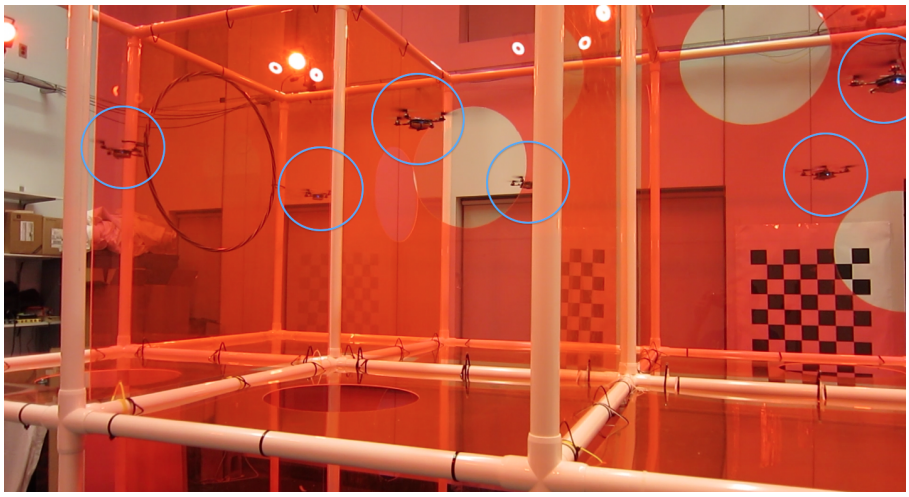


Figure 5.12: Six robots flying in the structure used for experimentation.

Figure 5.14 shows the spherical error probable (SEP) for a single robot tracking the original trajectory, the SEP for a single robot tracking a refined trajectory, and the SEP for 6 robots tracking refined trajectories. These plots clearly demonstrate the quadrotor's improved accuracy tracking these refined trajectories.

Robot states are tracked using a Vicon motion capture system. The material choice for the structure was carefully chosen to ensure localization performance anywhere inside

[1] `http://youtu.be/DRJPgOyN2so`

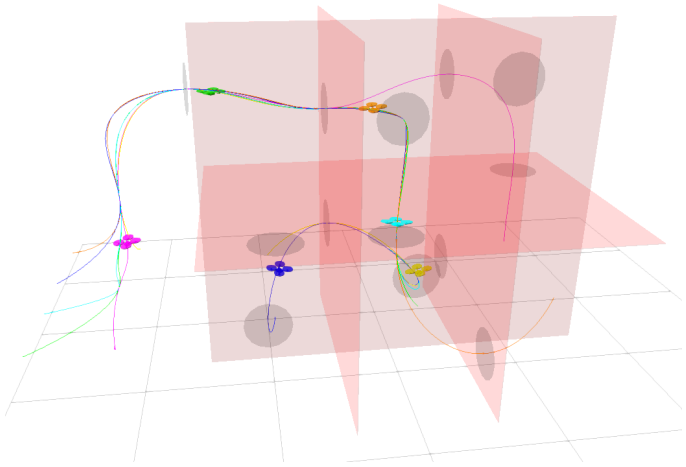


Figure 5.13: Visualization of actual experiment with refined minimum snap trajectories being tracked for 6 robots in confined structure. Video available online[1].

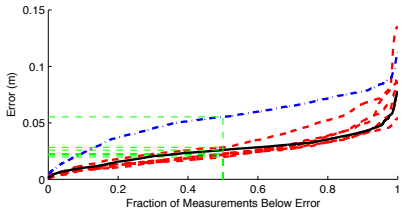


Figure 5.14: Spherical Error Probable (SEP) for trials with 1 robot following an unrefined trajectory (blue, dash dot), 1 robot following a refined trajectory (black solid), and 6 robots tracking refined trajectories as a team (red dashed). Notice that the refined trajectories give substantially lower error. Additionally, the initial trajectory takes 6 times longer than a refined trajectory to complete as it is frequently stopping and starting.

or outside of the structure. All experiments are computed on an external computer which is running control code to maintain the quadrotors in flight, as well as computing the trajectories as a separate process. Code is mostly written in MATLAB. Construction of the graph with over 1000 vertices generally takes less than 100 ms. Path planning for $N = 6$ robots reliably takes less than 1 ms. Assignment takes less than 10 ms. Prioritization generally takes 5 ms for 6 robots. Trajectory refinement using minimum snap trajectory generation requires between $5 - 10$ ms per robot, depending on the length of trajectory. Computing time offsets requires a large number of collision checks of ellipsoids, and therefore takes between $10 - 100$ ms for 6 robots depending on the length of trajectories. Thus, the total computation time is typically just under 0.2 s to generate dynamically feasible trajectories for 6 quadrotors through a tight space in MATLAB, however this could be optimized substantially to get better performance.

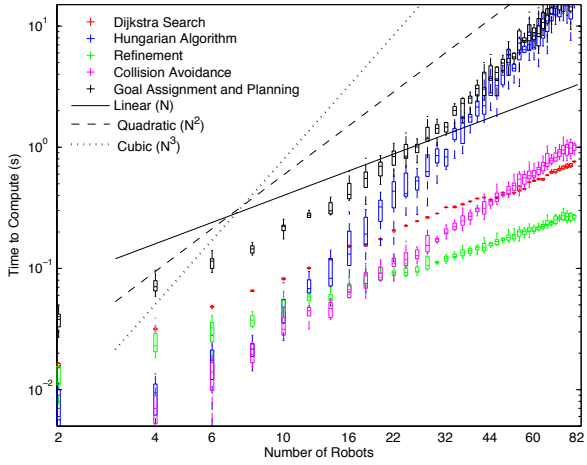


Figure 5.15: 100 simulation trials for each value of N to demonstrate time duration spent computing each stage of the algorithm applied to quadrotors. Note that both the planning (red) and refinement (green) grow roughly linearly in the number of robots and collision avoidance (pink) grows roughly quadratically. As the number of robots grows larger, the $\mathcal{O}(N^3)$ complexity of the Hungarian algorithm begins to dominate.

A much larger environment, similar to the maze-like structure in 5.11, was modeled in simulation to test the computational requirements as the number of robots grows much larger than the current experimental space allows. This simulated graph has $|V| > 10^4$, $|E| > 6 \times 10^5$, which for $N < 100$ are roughly constant in the number of simulated robots. The times of computation for each stage of the algorithm are displayed in Fig. 5.15.

This plot confirms the complexity analysis as a function of the number of robots in 5.2.8, where as the number of robots increases, solving the task assignment grows roughly cubically and begins to dominate the computation time at about 16 robots for the given parameters of the simulation. Of course, as $|V|$ or $|E|$ increase, the Dijkstra search will contribute more, moving the crossover of graph search and task assignment to higher N.

5.5 Chapter Summary

This chapter presents a complete trajectory generation method for a team of robots with complex dynamics. In-depth proofs of completeness and collision avoidance are included. The concept of trajectory refinement allows us to incorporate some robot dynamics while planning in a low dimensional space. A number of clarifying examples are provided and finally, the algorithm is applied to a team of quadrotor aerial vehicles.

Chapter 6: Multi-Robot Routing

The Multi-Robot Routing (MRR) problem seeks to safely and efficiently utilize a team of robots to visit a large number of goal locations. MRR is a common problem in autonomous exploration, surveillance, first response, and search and rescue. Consider a team of robots searching every room in a building for humans in need of assistance. Not only must the solution be of high quality, but it must be computationally tractable since the team of robots must start as soon as possible. It is assumed that any robot can visit any goal in any order, but every goal must eventually be visited by a robot. The MRR is a natural extension of the Vehicle Routing Problem (VRP) as outlined in Section 2.5 by explicitly adding collision constraints. Unfortunately, even before considering collision avoidance, this problem reduces to the travelling salesman problem for point robots and as a result, generating optimal plans is NP-Hard.

Already, Chapter 5 presents a powerful, complete method for finding trajectories for a team of interchangeable robots while avoiding collision. The GAP algorithm works very well when the number of robots is equal or approximately equal to the number of goal locations. However, GAP can yield quite poor solutions when a team of robots is tasked to visit a very large number of goal locations.

This chapter presents a polynomial time approximation algorithm for assigning and ordering N robots to visit M goals and is particularly relevant to the case where $M \gg N$. The proposed algorithm produces solutions that have maximum completion time no more

than 5 times the cost of the optimal solution. While the algorithm presented does not explicitly consider collision avoidance, some simple modifications ensure the solution incorporates collision avoidance. Handling collision in this way does not guarantee bounded time-optimal solutions, but does preserve the min-max distance property. Fortunately, due to its construction, the collision free approach tends to yield solutions with limited robot-robot interactions and will often preserve the time-optimal bound.

This problem of allocating robots to visit spatially distributed tasks is common in the operations research and multi-robot planning communities. There are numerous exact and approximation algorithms with a wide range of assumptions made about the solutions. The most basic problems are the Multiple Traveling Salesman Problem (MTSP) [7] and the Vehicle Routing Problem (VRP) [18]. Typically, solutions to the VRP seek to minimize the total distance traveled by all robots. With this cost function, it is possible that most robots travel a relatively short distance and a few agents must travel much further to ensure all locations are visited, unacceptable for time critical missions. Instead, minimizing the maximum distance traveled by an agent reduces completion time and is referred to as the Minimum Maximum Vehicle Routing Problem [72]. Another variant when robots are not necessarily beginning at a common starting location is defined as the Multiple Depot Vehicle Routing Problem [60]. In the MDVRP, agents can operate out of up to N unique starting locations with each agent returning to its original depot. This makes the MDVRP more suitable than the basic VRP for robotics since in applications in unstructured environments, robots typically do not work out of a common centralized facility. Allowing the robots to work out of unique locations also allows the system to replan based on updated goal information without requiring robots to return to the depot.

Therefore, the variation of the VRP that most closely resembles the work in this chapter is the Minimum Maximum Multiple Depot Vehicle Routing Problem (MMMDVRP) [9].

Another related problem is the orienteering problem, also known as the selective travelling salesperson problem, the maximum collection problem, and the bank robber problem. This is defined as follows: given time T, navigate to as many of M specified points as possible. [92]. Similarly, the Team Orienteering Problem (TOP) uses N agents to visit as many of the waypoints as possible in the allotted time.

An important result utilized in this chapter is an approximation algorithm for the min-max N path problem. This problem seeks to find at most N paths through all goal locations and [3] proposes an algorithm with a suboptimality bound of 4. However, this problem simply finds paths through goal points and does not include planning for robots at multiple starting locations. To bridge this gap, this chapter solves the assignment of robots to sequences which minimizes the maximum cost.

After preliminaries in Sect. 6.1, this chapter states the Multi-Robot Routing problem definition in Sect. 6.2. Section 6.3 presents Algorithm 4 that is shown to be a complete, polynomial time approximation algorithm to the MMMDVRP. Some refinement strategies are presented in Sect. 6.4 that often improve the solution quality. Next, Sect. 6.5 details how to avoid collisions between robots. Finally, Sect. 6.6 presents simulation results.

6.1 Preliminaries

In this work, N robots are tasked to visit M goal states $\gamma = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M\}$ where typically $M \gg N$. The initial state of robot i is $\mathbf{x}_i$, where $\sigma = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.

Much like in Chapter 5, an underlying graph $G_V = (V, E)$ is constructed to be used for

planning robot actions to reach goal locations. This graph can be constructed in either a stochastic or deterministic manner, but in this chapter, G_V is required to be undirected. The cost $C(v_1, v_2)$ is the cost, or travel time, of navigating from v_1 to v_2. In this chapter, robots are assumed to move at constant velocity such that the cost is equivalent to the navigation distance between vertices and therefore the function C respects the triangle inequality. Then, define G_γ as the graph that contains all goal vertices γ and up to $\binom{M}{2}$ edges have corresponding feasible or safe (ignoring robot-robot collisions) finite length paths connecting the goal states.

Sequence S_i is the list of goal vertices in the order they will be visited by a robot i. Since all goal locations must be visited by a robot, the elements in each sequence form disjoint sets with union equal to the set of all goal locations:

$$\{v \in S_1\} \cup \{v \in S_2\} \cup \ldots \cup \{v \in S_N\} = \gamma$$

$$\{v \in S_i\} \cap \{v \in S_j\} = \emptyset \quad \forall i \neq j$$

With slight abuse of notation, define the cost of traversing all edges in an ordered list of vertices S_i as $C(S_i)$. For example, if $S_1 = \{v_1, v_4, v_3\}$, and the cost of sequence 1 is defined as $C(S_i) = C(v_1, v_4) + C(v_4, v_3)$. Each element in a sequence can be referenced by position in the sequence using superscripts such that in the previous example, $S_1^{(1)} = v_1$.

A Hamiltonian path is defined as a path that visits each vertex in a set and in this case will be represented by a sequence. Define $\hat{H}(v_i, v_j, \ldots)$ as an approximation to the minimum cost Hamiltonian path to visit a set of vertices. $\hat{H}$ can be computed by doubling the Minimum Spanning Tree (MST) of all points in the set to visit, shortcutting, and removing an edge.This can be alternatively be computed using Algorithm 6 for higher quality solutions. However, this methods adds a substantial computation cost.

As in previous chapters, define the set of Z integers as $\mathcal{I}_Z \equiv \{1, 2, \ldots, Z\}$. It is assumed that robots have perfect actuation and have equal constant speeds. It is assumed that all robots are kinematic, constant velocity robots with noiseless sensing and actuation navigating in a known map.

6.2 Problem Statement

The cost of the optimal assignment of goals to robots and the sequencing of these goals is defined as:

$$J^\star = \underset{S_1, S_2, \ldots, S_N}{\text{minimize}} \quad \max_{i \in \mathcal{I}_N} \quad C(s_i, S_i^{(1)}) + C(S_i) \tag{6.2.1}$$

This optimal solution produces a result that has the minimum maximum cost solution to reach a sequence and navigate the sequence. Since the costs are assumed to be navigation time, this will produce a minimum time solution, also known as a minimum makespan solution.

The optimal assignment problem could be solved using brute force, however this would require an exponential number of operations. This is a result of the N^M possible assignments of goals to robots and for each of those assignments, the minimum cost Hamiltonian path needs to be found which has up to $M!$ permutations of the goals assigned. Instead of attempting to solve the NP-Hard routing problem, this chapter provides a polynomial time heuristic approach in Algorithm 4 that generates a solution with the maximum travel time of any robot no more than 5 times the optimal completion time $J^\star$.

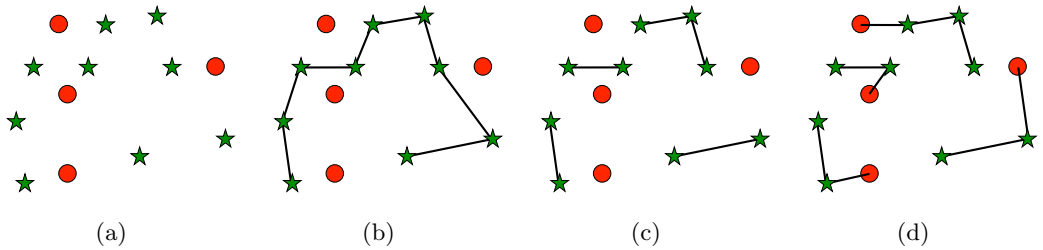


Figure 6.1: Figure 6.1(a) displays a set of 9 goal locations as green stars and 4 start locations as red circles. The MSF is constructed in Fig. 6.1(b). It happens in this example, that the MSF is also the result of doubling the MSF, removing an edge and shortcutting. Then in Fig. 6.1(c), this Hamiltonian path is split into N sequences. Finally in 6.1(d), the robots are each assigned to a sequences. Since this was successful, the next iteration in Algorithm 5 will have a smaller value of $\hat{J}$.

6.3 Approximation Algorithm

This section presents Algorithm 4, an approximation algorithm to the optimization in 6.2.1. Algorithm 4 provides an approximation ratio of 5, meaning that the solution produced will not have cost more than 5 times the optimal solution.

The basic flow of the method is as follows. For a given guess $\hat{J}$, Algorithm 5 attempts to return a solution with cost no more than $5\hat{J}$. If Algorithm 5 cannot find a solution, it will be shown that no solution exists with cost smaller than $\hat{J}$. This implies that $\hat{J} < J^\star$ and $\hat{J}$ must be increased to find a solution. Line 4 in Algorithm 4 performs a bisection search over $\hat{J}$ to find the smallest value of $\hat{J}$ that returns a solution. Lines 1-3 in 4 can be moved into 5, however the current placement yields improved computational performance through caching. A summary graphic of Algorithm 5 for a given guess $\hat{J}$ is displayed in Fig. 6.1. Now, each step in these algorithms will be discussed in greater detail.

Line 1 computes and caches the cost of a robot navigating from every start location to every goal state as well as the cost of navigating from every goal to every other goal

state. Since the graph is undirected, Dijkstra's algorithm [15] can be used to plan from a single goal to every other vertex in the graph with complexity of $\mathcal{O}(|E| + |V|\log|V|)$ [23]. This step is repeated for each goal state.

Line 2 constructs a Minimum Spanning Forest (MSF) of G_γ and line 3 sorts these edges into list F by the cost of a robot navigating the edge. These two steps can be done in one step using Kruskal's algorithm [40] for constructing MSFs with complexity bound $\mathcal{O}(M^2\log M)$. Note that F has at most $M-1$ edges.

Line 4 finds the minimum guess $\hat{J}$ that returns a solution to Algorithm 5. This can be performed using bisection search with upper and lower limits of 0 and the sum of all edge costs in the MSF plus the maximum distance each robot travels, respectively.

Algorithm 4 MultiRobotRouting

```
1: Compute paths to every goal state from every start and goal state, cache costs
2: Compute the Minimum Spanning Forest of G_γ
3: F ← list of edges in MSF(G_γ), sorted by cost
4: Find the minimum value Ĵ which returns a valid solution in Algorithm 5
5: if A valid solution was found then
6:     return Valid solution with minimum value Ĵ
7: else
8:     return No valid solution exists
```

Algorithm 5 begins by initializing U, the empty set of unordered sequences in line 1. Then, in lines 2-3 remove all edges longer than $\hat{J}$ from MSF(G_γ) and construct the MSF of the remaining graph. This is equivalent to the MSF of the original graph G_γ after removing all edges longer than $\hat{J}$. Computing the MSF has a total complexity of $\mathcal{O}\left(M\log(M)\right)$ since there are fewer than M edges in F.

Next, in lines 5-9, sequences to visit every vertex in the goal set are found, each of which has cost no larger than $4\hat{J}$. Line 6 computes the approximate minimum cost Hamiltonian

path visiting all vertices in the tree T_i as outlined in Sect. 6.1 where an Eulerian path through the graph can be found in linear time using Hierholzer's algorithm [28]. Line 7 computes k, the number of sequences required to visit every vertex in tree T_i such that each sequence has cost no more than $4\hat{J}$. Note that if $C(\hat{H}_i)$ is zero, this means that only one vertex is in the tree. A robot is still required to visit the vertex and therefore $k = 1$ in this case. The approximate Hamiltonian path is then split up into k sequences, each of which has cost no more than $4\hat{J}$ and then each of these sequences is added to the total list of sequences to visit U. Every input tree T_i is already a MST and each vertex is present only in one tree. Therefore lines 5-9 have total complexity bound of $\Theta(M)$.

Line 11 checks to see if there are a feasible number of sequences to visit. If there are more sequences than robots, the algorithm returns $J^\star > \hat{J}$. If, however, there are sufficient robots to visit the goals, then the algorithm finds the optimal assignment of robots to sequences.

Line 12 adds empty sequences to U to ensure it has N elements. Line 13 computes the cost matrix for assignment of robots to goals. Empty sequences have a cost of 0 as the robot does not need to move to visit every vertex in the sequence. Since the cost of navigating every robot to every sequence must be compared, this step has complexity bound of $\Theta(N^2)$.

Next in line 14, the bottleneck optimization problem is soved for the assignment of robots to sequences. This has complexity bound of $\mathcal{O}(N^3)$, but this can be slightly reduced in practice by removing edges in A larger than $\hat{J}$ and using the methods in [4].

Line 15 compares the maximum cost of reaching any sequence to the guess $\hat{J}$. If the maximum cost is greater than $\hat{J}$, then the algorithm returns $J^\star > \hat{J}$. Otherwise,

the algorithm returns the ordered list of sequences S, by rearranging the sequences in U according to the optimal bottleneck assignment. Finally, the algorithm returns this list of assigned sequences.

Algorithm 5 MinMax Cost Assigned Sequences($\hat{J}$)

$U \leftarrow \emptyset$
Define graph $G_{\hat{J}}$ as all goal vertices and edges in the MSF(G_γ) with cost less than $\hat{J}$
Compute MSF($G_{\hat{J}}$)

for Tree T_i in MSF($G_{\hat{J}}$) **do**
  Compute approximate minimum cost Hamiltonian path $\hat{H}_i \leftarrow \hat{H}(v \in T_i)$
  $k_i \equiv \max\left(\left\lceil \frac{C(\hat{H}_i)}{4\hat{J}} \right\rceil, 1\right)$
  Split $\hat{H}_i$ into k sequences, each of which has cost no larger than $4\hat{J}$
  Add these k_i sequences to set U

if $|U| \leq N$ **then**
  Add $(N - |U|)$ empty sequences to U
  $A_{ij} \leftarrow C(s_i, U_j^{(1)})$, cost to navigate robot i to the first vertex in sequence j
  Find bottleneck assignment of A
  if max cost of assignment $\leq \hat{J}$ **then**
    Reorder U into S using optimal assignment of robots to sequences
    return S

return FAILURE (this implies $J^\star > \hat{J}$)

6.3.1 Optimality Bound

By construction, any path returned has cost no more than $5\hat{J}$. This is a result of the fact that each sequence added to U has cost no more than 4 times $\hat{J}$ and the assignment ensures getting to the first vertex in a solution has cost no greater than $\hat{J}$.

Lemma 6.3.1. *Algorithm 5 never returns FAILURE unless $J^\star > \hat{J}$.*

Proof. There are two conditions which would result in the algorithm returning FAILURE. Case 1 occurs when there are more sequences in U than there are robots to visit sequences.

Case 2 arises when the cost of assigning any robot to a sequence is greater than $\hat{J}$. In each of these cases, it will be shown that the algorithm will never return FAILURE unless $J^\star > \hat{J}$.

Case 1. $|U| > N$. This proof of correctness is a direct result of Lemma 17 presented in [3], but will be outlined here. Assume the opposite that $|U| > N$ and $J^\star \leq \hat{J}$. By construction, each tree in the MSF is separated from every other tree by a distance of at least $\hat{J}$ since any larger value is removed from the MSF. In this case, $J^\star \leq \hat{J}$ and therefore, each tree can be considered individually since any edges connecting trees will by definition be larger than $\hat{J}$ and therefore larger than $J^\star$.

The optimal solution for tree i has connected $k_i^\star$ paths. These optimal paths for tree i can be connected into a spanning tree with edges each having cost no more than $\hat{J}$. The total cost of the connecting edges is no greater than $(k_i - 1)\hat{J}$ and by extension has cost less than $k_i J^\star$. Therefore this spanning tree has cost no greater than $2k_i J^\star$. Each edge of this tree can be doubled and every vertex can be visited with cost no greater than $4k_i J^\star$. Since this is the maximum cost of any path by construction, there is no possible way that $|U| > N$ while simultaneously $J^\star \leq \hat{J}$, leading to a contradiction. Therefore, the algorithm will never incorrectly return FAILURE due to $|U| > N$.

Case 2. $\max_i C(s_i, S_i^{(1)}) > \hat{J}$

It is not possible for the algorithm find $\max_i C(s_i, S_i^{(1)}) > \hat{J}$ when in truth $J^\star \leq \hat{J}$. Each vertex must be visited in the optimal solution and it is clear that every vertex in γ can be visited by some robot with cost no greater than $\hat{J}$. The bottleneck assignment used explicitly minimizes the maximum cost of this assignment.

□

Theorem 6.3.2. *Algorithm 4 either correctly returns that there is no solution to the robot routing problem or a solution that has cost of no more than* $5J^\star$.

Proof. For any sequences returned, $C(S_i) \leq 4\hat{J}$ is clear from construction. Similarly, the cost of navigating robot i to sequence S_i returned by Algorithm 5 is less than $\hat{J}$.

By performing binary search over $\hat{J}$ and utilizing the correctness property in Lemma 6.3.1, any returned solution from Algorithm 4 has the property that $\hat{J} \leq J^\star$. Therefore, if a solution is returned, the maximum cost of any robot reaching the assigned sequence plus the maximum sequence cost is no more than $5J^\star$ since the maximum cost to reach any sequence is no more than $J^\star$ and the maximum cost of any sequence is $4J^\star$. Therefore this algorithm has an approximation ratio of 5. □

6.3.2 Completeness

Assuming the underlying graph G_V is dense enough, Algorithm 4 will find a solution if a solution exists. This is due to the fact that $\hat{J}$ will continue to increase until reaching the sum of all edges in the graph plus the maximum cost to reach any vertex. At that point, the minimum spanning forest connects all goal vertices that can be visited by one robot. If each tree is able to be visited, regardless of cost, it will have a successful assignment.

6.3.3 Computational Complexity

This section will analyze the computational complexity for the approximation algorithm in Algorithm 4.

Lines 1-3 in Algorithm 4 have complexity $\mathcal{O}\left(M\left(|E| + |V|\log|V|\right) + M^2 \log M\right)$ as dis-

cussed previously.

The search to find the minimum feasible value of $\hat{J}$ can be reformulated to be strongly polynomial by replacing the bisection search with binary search over all possible breaking points for $\hat{J}$. Finding minimum maximum sequences constitute $\binom{M}{2}$ possible breaking points (see [3] for further detail) and the costs to navigate every robot to each goal introduce an additional NM breaking points. Therefore, Line 4 in Algorithm 4 calls Algorithm 5 a maximum of $\mathcal{O}(\log M)$ times.

Algorithm 5 is dominated by computing the $\mathcal{O}(N^3)$ bottleneck assignment [4] and computing the MSF with fewer than M edges. The overall complexity of Algorithm 5 is $\mathcal{O}(N^3 + M \log M)$.

To conclude, the overall complexity of Algorithm 4 is: $\mathcal{O}(M(|E| + |V| \log |V|) + (N^3 + M^2)\log M)$.

6.4 Refinement

This section discusses a number of modifications that typically result in higher quality solutions. However, these do not reduce the optimality bound and may add additional computational cost.

The first improvement will be discussed in detail in 6.4.1. After final assignment of robot to goal set, it is advantageous to replan $\hat{H}$ using Algorithm 6 to incorporate the initial conditions into the planning.

A second improvement can be made when performing the bottleneck assignment. It is reasonable to not only search over the minimum maximum value of the cost of assignment, but rather the minimum maximum value of cost of assignment plus the cost of the segment.

The success condition must also be changed such that the total cost of a robot getting to and visiting all goals in the assigned sequence is less then $5\hat{J}$. This does not not change the proof for correctness in Theorem 6.3.2.

Finally, in line 13 of Algorithm 5, the cost can be chosen as the minimum cost to navigate to either the first or last vertex. If the last vertex is closer to a robot, the sequence ordering can be reversed due to the undirected graph assumption.

6.4.1 Sequence Refinement

With slight abuse of notation, let $H^{\star}(s_i, S_i)$ be the minimum cost sequence beginning at s_i and visiting every vertex in S_i. While finding minimum cost Hamiltonian paths is known to be NP-Hard, Algorithm 6 finds a suboptimal sequence $\hat{H}(s_i, S_i)$ and is based on the Christofides Algorithm [11], which finds a bounded suboptimal solution to the TSP that has cost no more than $3/2$ times the optimal cost of the TSP. Lemma 6.4.1 proves that the suboptimality of this algorithm is bounded by a constant factor of 3/2 and this reordering of vertices will tend to improve performance over doubling each edge in the MST. For ease of reference, the subscripts for s_i and S_i are omitted in the algorithm and for the remainder of this section.

Algorithm 6 begins in line 1 by generating the graph G_S where vertices in G_S are those in S. Since one robot has been assigned to every goal in the sequence S, the graph is connected. Adding the fact that the original graph is undirected, this graph must be complete.

The graph G^O is constructed of all vertices in the MST with odd connectivity in line 2. It can be shown that the number of vertices in G^O is even (see [11]). The starting

location s is then added to the graph G^O with edge costs equal to the cost of visiting each of vertex in the graph. Dummy vertex d is added in lines 4-5, preserving an even number of vertices in G^O. The dummy vertex is selected to have infinite cost to the start vertex and zero cost to every other vertex. This ensures the starting robot will be matched to exactly one non-dummy vertex.

A minimum cost perfect matching [12] is found for the vertices in G^O in line 6. The dummy vertex will each be matched with a vertex other than the start vertex. The Eulerian path $\hat{H}$ is constructed in line 7 such that each edge in both the MST and in the perfect matching are visited once. The dummy vertex and duplicate vertices are removed in line 8. The computational cost of this algorithm is dominated by the cost for finding the minimum perfect matching, which has bounded complexity of $\mathcal{O}(|S|^3)$ for complete graphs [44].

Algorithm 6 SequenceRefinement $\hat{H}(s, S)$

1: Generate complete graph G_S from the set of all vertices in S
2: $G_O \leftarrow$ the set of goal vertices with odd connectivity in MST(G_S)
3: Add robot start s to G_O
4: $d \leftarrow$ dummy vertex with zero cost to goal vertices, infinite cost to s
5: Add d to G_O
6: $M^\star \leftarrow$ minimum perfect matching for G^O
7: $\hat{H} \leftarrow$ Eulerian path that begins at s, then traverses every edge in MST(G_S) $\cup$ $M^\star$
8: Remove d and duplicate vertices from $\hat{H}$
9: **return** $\hat{H}$

Lemma 6.4.1. *Algorithm 6 has approximation ratio 3/2:* $C(\hat{H}(s, S)) \leq \frac{3}{2} C(H^\star(s, S))$

Proof. This proof follows directly from [11] but adds a dummy vertex and only connects to the start vertex once.

It is clear that the sum of the cost of the edges in the minimum spanning tree is not

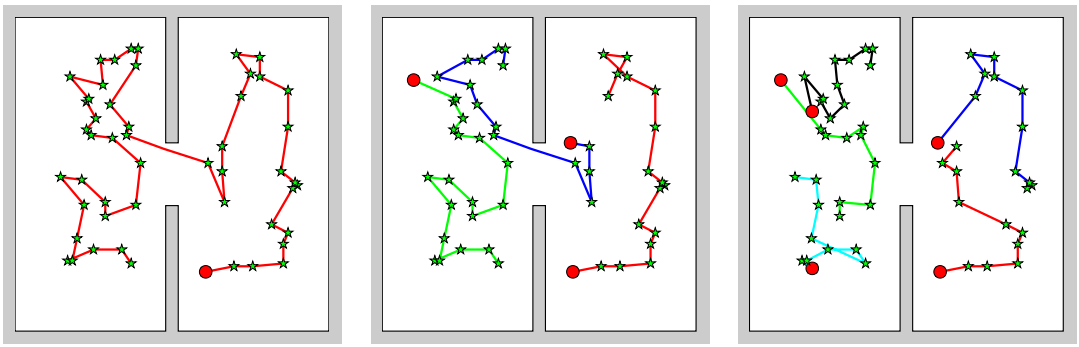
greater than the optimal path $C(\text{MST}(G_S)) \leq C(H^\star)$. Additionally, $C(M^\star) \leq \frac{1}{2}H^\star$ using the same logic as in [11]. After removing dummy vertices, but before removing duplicate vertices, the heuristic path $\hat{H}$ traverses each edge in $M^\star$ and $\text{MST}(G_S)$ once resulting in $C(\hat{H}) = C(\text{MST}(G_S)) + C(M^\star) \leq \frac{3}{2}C(H^\star)$. Since the graph respects the triangle equality, removing duplicate vertices will only reduce the cost of $C(\hat{H})$. □

6.5 Collision Avoidance

This section presents a method to ensure collision avoidance between robots for a orthogonally connected regular grid with grid size greater than robot diameter $2R$. Additionally, assume each robot moves one grid cell per unit time. It should be noted that the results in this section do not have time-optimal solutions, but do preserve the minimum-maximum distance optimality bound. In sufficiently open spaces, the minimum-maximum distance solution will tend to be close to the time-optimal solution.

First, predict the desired motion of each robot for one time step in the future. Between the current time step and the prediction, determine if two robots are attempting to swap goal states. In the event of such a crossing, robot exchange their sets of remaining goals. This strategy will decrease the distance traveled by either robot by at least 1 grid cell length.

Next, in the prediction step, check for multiple robots occupying the same cell. In this case, examine each robots' list of remaining goals to visit. If there is a robot occupying the cell at the current time, it exchanges sequences with the robot that has the highest remaining cost sequence. If there is a robot currently in the cell, it is guaranteed to move out of the cell. The robot attempting to enter the cell with highest remaining cost is



(a) $N = 1$, $M = 50$, $\hat{J} = 10.03$ (b) $N = 3$, $M = 50$, $\hat{J} = 3.35$ (c) $N = 5$, $M = 50$, $\hat{J} = 2.03$

Figure 6.2: Simulated trials with randomly generated consistent goal positions and varying N. Red circles are robot start locations and green stars are goal locations. The listed $\hat{J}$ value is the smallest value of $\hat{J}$ to return a solution for the problem. Notice how different the solutions are with different numbers of robots. This is a result of the refinement in Sect. 6.4.

allowed to enter and all other robots remain stationary. Using this rule, the maximum distance traveled by any robot does not increase.

These simple rules ensure the maximum distance traveled by a robot will not increase and collisions will be avoided. Additionally, these rules ensure progress is always being made by at least one robot and the system will eventually complete successfully.

6.6 Simulation Results

This section presents some simulation results using Algorithm 4. Figure 6.2 shows some basic results for small scale randomly generated problems.

Figure 6.4 presents the computation time trends as a function of M for a large number of trials with 10 robots. This figure nicely demonstrates the trends in computational time match the predictions from Sect. 6.3.3. In both the figure and prediction, computing the MSF has the largest growth with respect to increasing M. One deviation from expectation is the fact that refinement is expected to grow cubically in M. However, the simulations

use Blossom 5 [38], a state of the art minimum weight perfect matching solver and this cubic growth is only evident when thousands of goals are used. Note that simulations for 10 robots visiting 200 goals reliably takes less than 5 seconds total computation time.

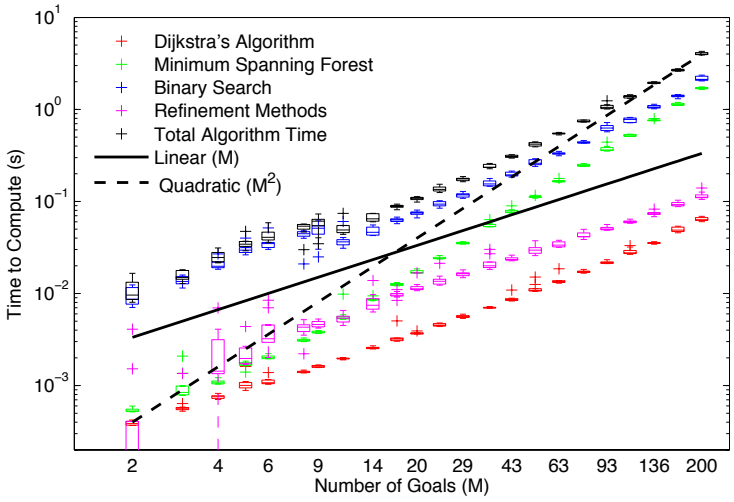


Figure 6.3: 100 simulation trials for each value of M with $N = 10$ to demonstrate computational growth. Note that both the planning (red) and refinement (pink) grow roughly linearly in the number of robots. The binary search of Algorithm 4 grows superlinear in M, but sub-quadratic as expected. Computing the MSF has expected complexity $\mathcal{O}(M^2 \log M)$ and begins to dominate computation time above $M = 200$.

Figure 6.5 demonstrates that the collision avoidance modification is sufficient to ensure collision avoidance for all robots.

6.7 Chapter Summary

This chapter presents a centralized method for generating sequences of goals to visit such that all goal locations are visited by point robots in no more time than a constant factor times the optimal. The computational complexity is shown to be polynomial in

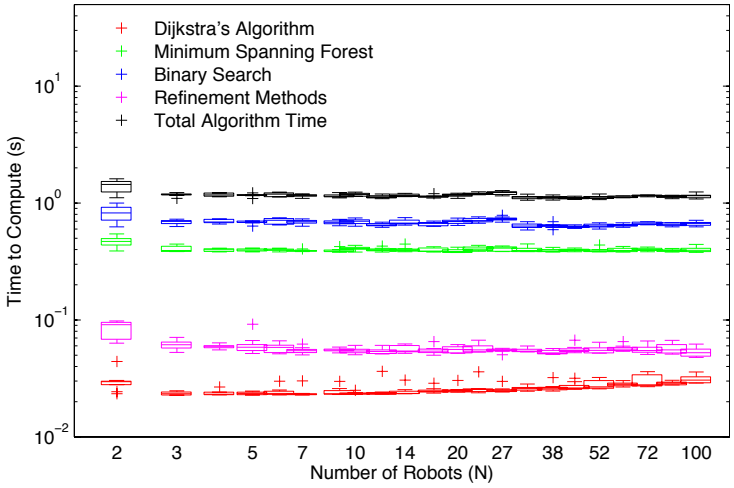
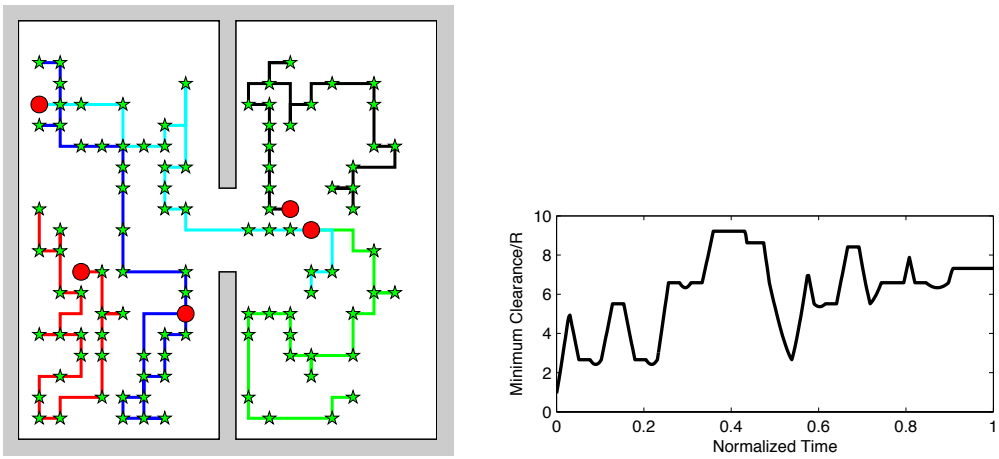


Figure 6.4: 100 simulation trials for each value of N with $M = 100$ to demonstrate computational growth. In this example, all steps have about constant computation time with respect to N. The $\mathcal{O}(N^3)$ expected growth for the binary search is not visible at such small values of N.

the number of robots and the number of goals. Then, a collision avoidance scheme was designed to ensure all robots safely reach their goal location for a simplified robot. Finally, a set of simulation results are presented to demonstrate the efficiency of the method.



(a) (b)

Figure 6.5: Figure 6.5(a) shows an a set of initial conditions to visit. Despite the high degree of interaction, it can be seen in 6.5(b) that no two robots ever have negative clearance and therefore avoid collisions.

Chapter 7: Conclusion

7.1 Contributions

This dissertation presents a number of novel solutions for multi-robot planning and control. First, the decentralized formation control approach presented in Chapter 3 generates a dynamically feasible safe trajectory for every robot in the team and is robust to individual robot failure. Then, robot interchangeability is utilized in Chapter 4 to safely plan trajectories for teams of robots in obstacle-free environments and scales well computationally as the number of robots increases. Next, an extension of these methods in Chapter 5 allows the robots to generate safe trajectories without collision while preserving completeness and optimality for dozens in a few seconds on a standard laptop computer. Finally, Chapter 6 further extends these methods to allow a team of robots to visit a very large number of goal locations with bounded suboptimality. Simulation and experimental results are presented throughout the dissertation to validate the theory.

Specifically, this dissertation quantifies the benefits of interchangeability of robots in a team if the mission allows it. The safety of the robots and nearby humans depends on the collision avoidance of teams. By utilizing the interchangeability of individual robots, this work demonstrates that a safe, complete planner can be developed that is computationally tractable and complete. These strong guarantees of completeness, collision avoidance, and polynomial complexity while providing optimality bonds is a necessity before teams

of robots will be utilized to the fullest extent.

Interchangeability of robots is acceptable for tasks where each robot is gathering information such as in search and rescue or exploration missions. A natural platform to use in these missions are aerial robots since these robots can quickly fly over any terrain and reach areas to inspect quickly. Application of these planning algorithms to a team of aerial robots operating outdoors requires a large degree of integration of sensing, perception, localization, mapping, and communication. This integration has begun in [58] by applying the methods developed in this dissertation to a team of quadrotor robots outdoors navigating using a combination of GPS and vision to quickly gather information.

7.2 Future Work

The approaches detailed in Chapters 4-6 of this dissertation do not incorporate sensing or robot noise and assume robots can precisely follow the plans generated for them. It might be beneficial to model the noise of the robots such that more informed planning algorithms and control laws can be used. For example, it is challenging for robots to localize well in long corridors using laser scanners since the world looks identical regardless of the location of the robot. If instead, the robots were to account for this difficulty in localization in the environment and plan an alterante route to avoid the hallway, individual robots should have higher quality position estimates. There are a number of existing approaches to incorporate state dependent noise including such as planning in belief space [71] that could be incorporated into the work in this dissertation.

The formation control approach presented in Chapter 3 is able to reject noise in actuation and sensing through the closed loop control policy. However, it does not use

the information quality in determining which robots have more useful information in their trajectories being communicated. It would be particularly interesting to investigate how to change dependency coefficients $\mathbf{D}$ in Chapter 3 as a function of robot errors.

This work in Chapters 4-6 only permit teams of fully interchangeable robots and a related topic that might be useful when this assumption is not valid is the relaxation of this total interchangeability. This idea has been studied as the k-Color Problem in [79] where the robots and goals fall into a number of classes ranging from fully labeled to fully interchangeable. Thers is no algorithm that is able to incorporate multiple classes of robots while allowing for resolution completeness, trajectory refinement, or bounded optimality solution. The algorithm presented in Chapter 5 uses a decoupled algorithm and it can be shown relatively easily that a fully decoupled algorithm is not sufficient to if the interchangeable assumption is removed. Instead, another approach must be investigated to incorporate a number of classes of robots while retaining their optimality bounds, completeness, and polynomial time complexity.

The work in Chapter 6 requires robots to remain stationary at goal locations for the planning to be successful. This is a result of only admitting one goal state in the graph for each goal location specified and this state is required to have zero velocity to ensure the completeness property. It may be possible to relax the zero velocity constraint at each goal location.

Finally, another interesting research avenue would be the investigation of additional refinement steps in Chapter 6. Currently, the segments are divided using a simple process of attempting to minimize the largest segment. It may be possible to incorporate the locations of robots and the assignment of robot to goals when dividing segments. This

would provide a better solution in practice, however would still not improve the bound on suboptimality.

Appendix A: Quadrotor Basics and Control

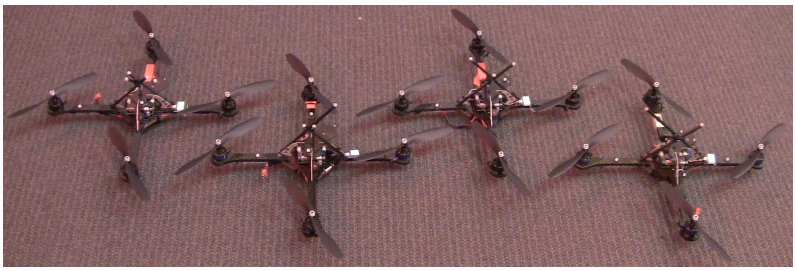


Figure A.1: A team of four Ascending Technologies [5] commercial hummingbird quadrotors used in experimentation.

Quadrotors are mechanically simple flying robots with the ability to take off and land vertically, carry small to medium payloads, hover in place, and perform complex aerobatic maneuvers. The ability to fly means that these robots can simply travel above difficult terrain that would halt a ground based platform. While any flying platform has the means to bypass rough ground in this manner, fixed wing robots require a minimum forward velocity and therefore have substantial difficult functioning in tight spaces and specifically are extremely tough to use indoors. While helicopters can also fly with similar functionality to multi-rotor craft, they have complex and expensive swash plates and linkages. Multi-rotor craft have a very simple design with the only moving parts being their propellers directly connected to motors without any gear train. A multi-rotor craft with fixed rotors requires at least four rotors if the frame is stationary and therefore, a quadrotor is one of the simplest, yet most versatile robots ever envisioned.

This substantial feature list of the quadrotor robots is not without its draw backs.

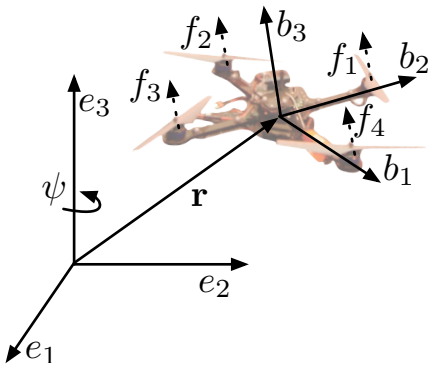


Figure A.2: A quadrotor with world and body frames shown. The position and orientation of the robot in the global frame are denoted by x and R, respectively. The quadrotor uses the thrust generated by each propeller (along the body-fixed z-axis, b_3) to generate forces and moments for control.

Notably, controlling the robots can be challenging and has only been practical in recent years due to the rapid increase in small, lightweight sensing and computing. In addition, current battery technology limitations mean that quadrotors are limited by their short flight times of generally under 30 minutes on a single battery charge.

This chapter presents a dynamic model of a quadrotor and a controller which allows a single robot to track specified 3-D trajectory. This control is utilized throughout the remainder of this dissertation to control each robot individually.

It is well-known that the quadrotor is underactuated and differentially flat [61]. By using nonlinear controllers with no reliance on linearized models of the quadrotor's dynamics, these robots are able to follow trajectories that require large roll and pitch angles that produce high accelerations in the horizontal plane. Figure A.2 depicts the reference frames used by the control laws presented in this chapter.

The four rotor inputs allow us to specify the force along the body-fixed z-axis and the three moments in the body-fixed frame. Accordingly, the four output variables chosen are

the three spatial components and the world yaw angle to specify the desired trajectory in the time interval $[t_0, t_f]$:

$$\mathbf{x}_{\mathrm{d}}(t) : [t_0,\, t_f] \to \mathbb{R}^3 \times SO(2) \tag{A.0.1}$$

The controllers achieve attitude stabilization in $SO(3)$ with a basin of attraction that covers almost all of the rotation group [45, 46]. The specification of the trajectory restricted to the subgroup $\mathbb{R}^3 \times SO(2)$ allows each robot to choose its roll and pitch angle to track the specified trajectory. This specification is also practical from the standpoint of commanding the robot as it is quite natural to specify the desired position trajectory and the heading (yaw) along the robot trajectory.

As the control input to the quadrotor first shows up in controlling the fourth derivative of the spatial outputs, the quadrotor is a fourth order system. This means that any trajectory that the quadrotor can track is at least $\mathbb{C}^3$ smooth (with the notable exception of yaw, which must be $\mathbb{C}^1$ smooth). As snap is the fourth derivative of position, it is logical to plan trajectories which minimize snap [53]. For this reason, piecewise-smooth polynomial functions of order p are used to represent trajectories:

$$\mathbf{x}_{\mathrm{d}}(t) = \sum_{k=0}^{p} \boldsymbol{\beta}^k t^k \tag{A.0.2}$$

These trajectories must also satisfy constraints on state and input variables such as bounded rotor speed.

Consider a quadrotor with mass m and rotational inertia $J \in \mathbb{R}^3$. Define the position and rotation of the vehicle in the inertial frame as $x \in \mathbb{R}^3$ and $R \in SO(3)$, respectively. The angular velocity of the vehicle, $\Omega \in \mathbb{R}^3$, is given as:

$$\dot{R} = R\hat{\Omega}$$

where the operator $\hat{\cdot}$ is defined such that $\hat{x}y = x \times y$ for all x, $y \in \mathbb{R}^3$. Given that the i^{th} propeller generates the thrust output f_i as a function of propeller rotational speed, the dynamic model of the vehicle follows:

$$m\ddot{x} = (fR - mg)e_3$$

$$J\dot{\Omega} + \Omega \times J\Omega = M$$

with $e_3 = [0,\, 0,\, 1]^{\mathrm{T}}$, $M = [M_1,\, M_2,\, M_3]^{\mathrm{T}}$, and

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d & 0 & -d & 0 \\ 0 & d & 0 & -d \\ -c & c & -c & c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

where d is the distance from the robot center of mass to the rotor and c is an aerodynamic drag term that relates differences in propeller speed to yawing moment about the body z-axis.

Following the attitude stabilization approach proposed in [46], we design the force and moment inputs, f and M_i, based on the desired input $\mathbf{x}_{\mathrm{d}}(t)$:

$$\begin{aligned} f &= (-k_x e_x - k_{\dot{x}} e_{\dot{x}} + mge_3 + m\ddot{x}_{\mathrm{d}}) \cdot Re_3 \\ M &= -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega \end{aligned} \tag{A.0.3}$$

with the error terms defined as follows:

$$\begin{aligned} e_x &= x - x_{\mathrm{d}} \\ e_{\dot{x}} &= \dot{x} - \dot{x}_{\mathrm{d}} \\ e_R &= \frac{1}{2\sqrt{1 + \mathrm{tr}\left[R_d^{\mathrm{T}} R\right]}} \left(R_{\mathrm{d}}^{\mathrm{T}} R - R^{\mathrm{T}} R_{\mathrm{d}}\right)^{\vee} \\ e_\Omega &= \Omega - R^{\mathrm{T}} R_{\mathrm{d}} \Omega_{\mathrm{d}} \end{aligned}$$

and dropping the dependence on time for clarity of presentation. The operator $(\cdot)^{\vee}$ is the inverse of the $\hat{\cdot}$ operator such that $(\cdot)^{\vee} : so(3) \to \mathbb{R}^3$. The gains k_x, $k_{\dot{x}}$, k_R, and k_Ω are

selected to ensure stable performance. See [45, 46] for further explanation of the derivation of these error terms and proofs of stability and convergence of the control system to the desired inputs.

The above attitude stabilization approach operates in $SO(3)$ (as compared to the traditional Euler-angle parameterization approach [56]) and benefits from a stability basin of attraction that includes the full space of rotation matrices (excluding an exact inversion). Note that it is assumed the trajectory generation scheme plans trajectories with the constraint $f > 0$ and thus define R_d with respect to ψ_d such that $R_\mathrm{d} = [r_1, r_2, r_3]$ and

$$
\begin{aligned}
r_1 &= r_2 \times r_3 \\
r_2 &= \frac{r_3 \times [\cos\psi_\mathrm{d}, \sin\psi_\mathrm{d}, 0]}{\| r_3 \times [\cos\psi_\mathrm{d}, \sin\psi_\mathrm{d}, 0] \|} \\
r_3 &= \frac{-k_x e_x - k_{\dot{x}} e_{\dot{x}} + mge_3 + m\ddot{x}_\mathrm{d}}{\| -k_x e_x - k_{\dot{x}} e_{\dot{x}} + mge_3 + m\ddot{x}_\mathrm{d} \|}
\end{aligned}
$$

Additionally, the moment stabilization proposed in [46] includes higher-order inertial cancelation terms that are neglected due to their insignificant impact.

To find smooth trajectories that minimize an integral cost function consisting of time derivatives up to the l^{th} derivative of the trajectory, piecewise-smooth polynomial functions of order p are used over a series of time intervals. For simplicity of notation, this work will often replace the series by the use of a single polynomial function of order p.

To find the trajectory that minimizes the cost functional derived from the l_r^{th} derivative of position and l_ψ^{th} derivative of the yaw angle, consider the following optimization

program:

$$\begin{aligned}
\text{minimize} \quad & \int_{t_0}^{t_f} \left[\mu_{\mathbb{R}^3} \left|\left| \frac{d^{l_{\mathbf{r}}} x}{dt^{l_{\mathbf{r}}}} \right|\right|^2 + \mu_\psi \left(\frac{d^{l_\psi} \psi}{dt^{l_\psi}} \right)^2 \right] dt \\
\text{subject to} \quad & \mathbf{x}(t) = \sum_{k=0}^{p} \boldsymbol{\beta}^k t^k \\
& \left. \frac{d^g x}{dt^g} \right|_{t=0} = \mathbf{0} \quad g = 1, \ldots, l_{\mathbf{r}} \\
& \left. \frac{d^g x}{dt^g} \right|_{t=t_f} = \mathbf{0} \quad g = 1, \ldots, l_{\mathbf{r}} \\
& \left. \frac{d^g \psi}{dt^g} \right|_{t=0} = \mathbf{0} \quad g = 1, \ldots, l_\psi \\
& \left. \frac{d^g \psi}{dt^g} \right|_{t=t_f} = \mathbf{0} \quad g = 1, \ldots, l_\psi
\end{aligned} \tag{A.0.4}$$

where $\mu_{\mathbb{R}^3}$ and μ_ψ are constants that make the integrand non-dimensional. Due to the dynamics of the quadrotor noted previously, consider different derivatives of interest for each of the desired inputs (in particular $l_{\mathbf{r}} = 4$ for position components and $l_\psi = 2$ for yaw components). State and input constraints such as limits on angular rates and propellor thrusts can be expressed as algebraic functions of $\mathbf{x}$ and its derivatives, and therefore can be incorporated in this formulation as additional constraints. The above optimization is formulated as a quadratic program (QP) with initial conditions and derivative constraints defined as equality and inequality constraints as required. Further details of the methodology including the approach to determining the non-dimensional constants and using piecewise smooth polynomials are available in [53]. Note that the trajectory resulting from (A.0.4) is parameterized in terms of the coefficients $\boldsymbol{\beta}^k$, and only these are required to fully specify a trajectory.

Aerodynamic interactions of the quadrotors require additional considerations for the safety of the team. A robot flying below another robot is adversely affected by the

downwash from the higher robot and will be unable to track trajectories satisfactorily. For this reason, an ellipsoidal model of the quadrotor is used to ensure that the separation in the vertical direction is larger than the separation in the horizontal direction:

$$\sqrt{\left(\frac{x_i(t)-x_j(t)}{1}\right)^2+\left(\frac{y_i(t)-y_j(t)}{1}\right)^2+\left(\frac{z_i(t)-z_j(t)}{4}\right)^2} > 2R \quad \forall\, i,j,t$$

This condition means that for a robot to fly directly above another robot, the separation in the vertical direction must be at least $8R$.

Throughout this work, state estimation is not considered and therefore perfect state knowledge is assumed. The pose of the quadrotor is observed using a VICON motion capture system at 100 Hz [93]. The pose is numerically differentiated to compute the linear and angular velocities of the robot. Desired attitude commands are interpreted by the onboard attitude and body-fixed thrust controllers (A.0.3) operating on each robot's programmable embedded microprocessor and applied at an update rate of 1 kHz.

Bibliography

[1] Abdelkader Abdessameud and Abdelhamid Tayebi. Formation control of vtol unmanned aerial vehicles with communication delays. *Automatica*, 47(11):2383–2394, 2011.

[2] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Roland Siegwart, and Paul Beardsley. Image and animation display with multiple mobile robots. *The International Journal of Robotics Research*, 31(6):753–773, 2012.

[3] Esther M Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.

[4] Ronald D Armstrong and Zhiying Jin. Solving linear bottleneck assignment problems via strong spanning trees. *Operations research letters*, 12(3):179–180, 1992.

[5] Ascending Technologies, GmbH. `http://www.asctec.de`.

[6] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[7] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[8] Rainer E Burkard and Franz Rendl. Lexicographic bottleneck problems. *Operations Research Letters*, 10(5):303–308, 1991.

[9] John Carlsson, Dongdong Ge, Arjun Subramaniam, Amy Wu, and Yinyu Ye. Solving min-max multi-depot vehicle routing problem. *Lectures on Global Optimization. Fields Institute Communications*, 55:31–46, 2009.

[10] L. Chaimowicz, N. Michael, and V. Kumar. Controlling swarms of robots using interpolated implicit functions. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2487–2492, Barcelona, April 2005. IEEE.

[11] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[12] William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.

[13] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor. A vision-based formation control framework. *IEEE Trans. Robot. Autom.*, 18(5):813–825, 2002.

[14] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Trans. Robot.*, 17(6):905–908, December 2001.

[15] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[16] Robert Dougherty, Veronica Ochoa, Zachary Randles, and Christopher Kitts. A behavioral control approach to formation-keeping through an obstacle field. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 1. IEEE, 2004.

[17] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Trans. Robot. Autom.*, 17(6):947–951, December 2001.

[18] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[19] J.J. Enright and P.R. Wurman. Optimization and coordinated autonomy in mobile fulfillment systems. In *Workshops at AAAI Conf. on Artificial Intelli.*, volume 1, page 2, 2011.

[20] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Trans. Robot. Autom.*, volume 3, pages 1419–1424, 1986.

[21] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. 49(9):1465–1476, September 2004.

[22] Antonio Franchi, Carlo Masone, HH Bulthoff, and Paolo Robuffo Giordano. Bilateral teleoperation of multiple uavs with decentralized bearing-only formation control. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2215–2222. IEEE, 2011.

[23] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[24] Jesús García, Antonio Berlanga, José M Molina, Juan A Besada, and José R Casar. Planning techniques for airport ground operations. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 1, pages 1D5–1. IEEE, 2002.

[25] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[26] Y. Gu, B. Seanor, G. Campa, M. R. Napolitano, L. Rowe, S. Gururajan, and S. Wan. Design and flight testing evaluation of formation control laws. *IEEE Trans. Control Syst. Technol.*, 14(6):1105–1112, November 2006.

[27] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[28] Carl Hierholzer and Chr Wiener. Über die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6(1):30–32, 1873.

[29] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. 48(6):988–1001, June 2003.

[30] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding-horizon control of nonlinear systems. *Automatic Control, IEEE Transactions on*, 46(5):776–783, May 2001.

[31] M. Ji, S. Azuma, and M.B. Egerstedt. Role-assignment in multi-agent coordination. *Int. Journal of Assistive Robotics and Mechatronics*, 7(1):32–40, March 2006.

[32] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[33] T. Keviczky and K.H. Johansson. A study on distributed model predictive consensus. *Arxiv preprint arXiv:0802.4450*, 2008.

[34] Chang W Kim and Jose MA Tanchoco. Conflict-free shortest-time bidirectional agv routeing. *The International Journal of Production Research*, 29(12):2377–2391, 1991.

[35] Kap Hwan Kim, Su Min Jeon, and Kwang Ryel Ryu. Deadlock prevention for automated guided vehicles in automated container terminals. In *Container Terminals and Cargo Systems*, pages 243–263. Springer, 2007.

[36] Stephen Kloder and Seth Hutchinson. Path planning for permutation-invariant multirobot formations. *Robotics, IEEE Transactions on*, 22(4):650–665, 2006.

[37] KMel robotics. `http://kmelrobotics.com`.

[38] Vladimir Kolmogorov. Blossom v: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

[39] Nirup N Krishnamurthy, Rajan Batta, and Mark H Karwan. Developing conflict-free routes for automated guided vehicles. *Operations Research*, 41(6):1077–1090, 1993.

[40] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[41] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[42] G. Latouche and V. Ramaswami. Introduction to matrix analytic methods in stochastic modeling. *ASA-SIAM, Philadelphia*, 1999.

[43] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[44] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Dover Publications, 1976.

[45] T. Lee. Geometric tracking control of the attitude dynamics of a rigid body on SO(3). In *Proc. of the Amer. Control Conf.*, San Francisco, CA, April 2011.

[46] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). Atlanta, GA, December 2010.

[47] M Anthony Lewis and Kar-Han Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[48] Q. Lindsey, D. Mellinger, and V. Kumar. Construction with quadrotor teams. *Autonomous Robots*, 33:323–336, 2012.

[49] L. Liu and D. A. Shell. Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *Proc. of Robot.: Sci. and Syst.*, Los Angeles, CA, June 2011.

[50] Lantao Liu and Dylan A Shell. A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Robotics: Science and Systems*, 2012.

[51] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[52] Ryan Luna and Kostas E Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 294–300. AAAI Press, 2011.

[53] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2520–2525, Shanghai, China, May 2011.

[54] M. Mesbahi. On state-dependent dynamic graphs and their controllability properties. 50(3):387–392, March 2005.

[55] N. Michael, M.M. Zavlanos, V. Kumar, and G.J. Pappas. Distributed multi-robot task assignment and formation control. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 128–133, Pasadena, CA, May 2008.

[56] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP multiple micro UAV testbed. *IEEE Robot. Autom. Mag.*, 17(3):56–65, September 2010.

[57] Rolf H Möhring, Ekkehard Köhler, Ewgenij Gawrilow, and Björn Stenzel. Conflict-free real-time agv routing. In *Operations Research Proceedings 2004*, pages 18–24. Springer, 2005.

[58] Kartik Mohta, Matthew Turpin, Alex Kushleyev, Daniel Mellinger, Nathan Michael, and Vijay Kumar. Quadcloud: A rapid response force with quadrotor teams. In *International Symposium on Experimental Robotics (ISER)*, Marrakech and Essaouira, Morocco, June 2014.

[59] P. Molnár and J. Starke. Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behavior. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 31(3):433–435, 2001.

[60] Gabor Nagy and Saı d Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141, 2005.

[61] M. J. Van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. *Intl. J. Robust and Nonlinear Control*, 8(11):995–1020, December 1998.

[62] Martin Oellrich. *Minimum Cost Disjoint Paths under Arc Dependences. Algorithms for Practice*. PhD thesis, Universitätsbibliothek, 2008.

[63] P. Ogren, E. Fiorelli, and N. Leonard. Formations with a mission: stable coordination of vehicle group maneuvers. In *Proc. of Intl. Sym. on Mathematical Theory Networks and Syst.*, Notre Dame, IN, August 2002.

[64] Ian O'Hara, James Paulos, Jay Davey, Nick Eckenstein, Neel Doshi, Tarik Tosun, Jonathan Greco, Jungwon Seo, Matthew Turpin, Vijay Kumar, and Mark Yim. Self-assembly of a swarm of autonomous boats self-assembly of a swarm of autonomous boats into floating structures. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Hong Kong, 2014.

[65] R. Olfati-Saber and R. M. Murray. Distributed cooperative control of multiple vehicle formations using structural potential functions. In *Proc. of the IFAC World Congress*, Barcelona, Spain, July 2002.

[66] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. 49(9):1520–1533, September 2004.

[67] R. Oung and R. D'Andrea. The distributed flight array. *Mechatronics*, 21(6):908–917, 2011.

[68] Dimitra Panagou, Matthew Turpin, and Vijay Kumar. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach. In *Proc. IEEE Intl Conf. on Robotics & Automation*, Hong Kong, June 2014.

[69] Mike Peasgood, Christopher Michael Clark, and John McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *Robotics, IEEE Transactions on*, 24(2):283–292, 2008.

[70] ER Petersen and AJ Taylor. An optimal scheduling system for the welland canal. *Transportation science*, 22(3):173–185, 1988.

[71] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 2009.

[72] Chunyu Ren. Solving min-max vehicle routing problem. *Journal of Software (1796217X)*, 6(9), 2011.

[73] F. Rendl. On the euclidean assignment problem. *Journal of Computational and Applied Mathematics*, 23(3):257–265, 1988.

[74] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for quadrotor flight. In *International Conference on Robotics and Automation*, 2013.

[75] Kaspar Schüpbach and Rico Zenklusen. Approximation algorithms for conflict-free vehicle routing. In *Algorithms–ESA 2011*, pages 640–651. Springer, 2011.

[76] D.H. Shim, H.J. Kim, and S. Sastry. Decentralized nonlinear model predictive control of multiple flying robots. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 4, pages 3621–3626. IEEE, 2003.

[77] S. L. Smith and F. Bullo. Target assignment for robotic networks: Asymptotic performance under limited communication. In *Proc. of the Amer. Control Conf.*, pages 1155–1160, New York, July 2007.

[78] PT Sokkalingam and Yash P Aneja. Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, 23(1):27–33, 1998.

[79] Kiril Solovey and Dan Halperin. k-color multi-robot motion planning. *The International Journal of Robotics Research*, 33(1):82–97, 2014.

[80] P. Tabuada, G. J. Pappas, and P. Lima. Feasible formations of multi-agent systems. In *Proc. of the Amer. Control Conf.*, pages 56–61, Arlington, VA, June 2001.

[81] H. Tanner, G. J. Pappas, and V. Kumar. Input-to-state stability on formation graphs. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2439–2444, Las Vegas, NV, December 2002.

[82] M. Turpin, N. Michael, and V. Kumar. Trajectory design and control for aggressive formation flight with quadrotors. In *Proc. of the Intl. Sym. of Robotics Research*, Flagstaff, AZ, August 2011.

[83] M. Turpin, N. Michael, and V. Kumar. Decentralized formation control with variable shapes for aerial robots. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, St. Paul, MN, May 2012.

[84] M. Turpin, N. Michael, and V. Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33(1-2):143–156, 2012.

[85] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic Foundations of Robotics X*, pages 175–190. Springer, Boston, MA, June 2012.

[86] Matthew Turpin, Nathan Michael, and Vijay Kumar. Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2013.

[87] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

[88] Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of aerial robots. In *Robotics Science and Systems*, Berlin, Germany, June 2013.

[89] Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots (Under Review)*, 2014.

[90] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer, 2011.

[91] Jur van Den Berg, Jack Snoeyink, Ming C Lin, and Dinesh Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems*, volume 2, pages 2–3. Citeseer, 2009.

[92] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[93] Vicon Motion Systems, Inc. `http://www.vicon.com`.

[94] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, 1995.

[95] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3260–3267, San Francisco, CA, Sept. 2011.

[96] Glenn Wagner, Minsu Kang, and Howie Choset. Probabilistic path planning for multiple robots with subdimensional expansion. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2886–2892. IEEE, 2012.

[97] J. Werfel, D. Ingber, and R. Nagpal. Collective construction of environmentally-adaptive structures. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2007.

[98] Jingjin Yu and Steven M LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 4023–4028. IEEE, 2012.

[99] M. M. Zavlanos and G. J. Pappas. Distributed formation control with permutation symmetries. pages 2894–2899, New Orleans, LA, December 2007.

[100] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Trans. Robot.*, 23(4):812–816, 2007.