Decentralized Coordinated Motion for Robot Teams Preserving Connectivity and Avoiding Collisions

Anqi Li

CMU-RI-TR-17-14

May 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Keywords: Multi-Robot Systems, Decentralized Algorithms, Algebraic Topology

For my family and friends

Abstract

In this thesis, we consider the general problem of moving a large number of networked robots toward a goal position through a cluttered environment under constraints on network connectivity and collision avoidance. In contrast to previous approaches that either plan complete paths for each individual robot in the high-dimensional joint configuration space or control the robot group as a whole with explicit constraints on the group's boundary and inter-robot pairwise distances, we propose a novel decentralized online behavior-based algorithm that relies on the topological structure of the multi-robot communication and sensing graphs to solve this problem. We formally describe the communication graph as a simplicial complex that enables robots to iteratively identify the frontier nodes and coordinate forward motion through the sensing graph. This approach is proved to automatically deform robot teams for collision avoidance and always preserve connectivity. The effectiveness of our approach is demonstrated using numerical simulations. The algorithm is shown to scale linearly in the number of robots.

Acknowledgments

First and foremost, I would like to express my profound gratitude to my advisor Prof. Katia Sycara. Without her guidance I would never be exposed to the fantastic field of multi-robot system. Her patience, enthusiasm, vision and knowledge is invaluable to this thesis. I would also like to acknowledge my committee member Prof. Magnus Egerstedt from Georgia Institute of Technology for his very valuable comments on this thesis. It is a great pleasure working with all of you.

Next, my special thank goes to Sasanka Nagavalli and Wenhao Luo for consistently giving me their precious support throughout my Masters study – both inside and outside research. You always steered me in the right the direction whenever I feel lost.

Further, I would like to thank my lab mates Navyata Sanghvi and Ramitha Sundar, for their insightful comments and encouragement to my research. I am also grateful to Prof. Michael Lewis (University of Pittsburgh) and Prof. Nilanjan Chakraborty (Stony Brook University) who provided insight and expertise that greatly assisted my research.

Finally, I want to thank my parents for their endless support and encouragement throughout my life.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Networked decentralized multi-robot systems employ local communication and collaborative decision making to carry out a wide variety of large-scale applications such as search and rescue, exploration of unknown environments and environmental sampling. The objective of the research in this thesis is to design a coordination strategy for large groups of robots to navigate in a cluttered environment. We focus on three challenges that guiding large-scale robot team faces in real-world application. In the first, the system should be safe, in that the robots should not collide with one another or obstacles in the environment. In the second, the robots should remain connected while coordinating. In the third, the coordination strategy should be scalable as the number of robots in the system increases.

We study the problem of coordinating the motion of a large group of robots towards a goal region in a decentralized manner through a cluttered environment that contains narrow corridors and static obstacles, while avoiding collisions and ensuring connectivity. We propose to employ the topological structure of the multi-robot communication and sensing graphs to compute the robots' incremental movements at each time step so as to reduce the navigation complexity. This allows for interleaving planning and execution that naturally captures the changing graph topology of the moving robot team and restricts robot movements to preserve connectivity and avoid collisions in unknown environments.

In this paper, we propose a novel decentralized and behavior-based approach for a large group of robots moving in unknown environments with obstacles. Our approach deals with all of the aforementioned challenges at the same time. Inspired by [16] that drives robots for sensor coverage based on abstract simplicial complexes from algebraic topology [8], we reduce the dimensionality of our problem by formally describing the communication graph of the robot team as an abstract simplicial complexes that provides feasible frontier nodes for computing robots' incremental motions. In our approach only a subset of the robots move at each time step, hence the robots form a *lattice band* formation as is shown in Figure 4.1. The *lattice band* will autonomously deform according to the obstacle-filled environment. We prove that (a) the robots will never lose connectivity or collide with one another or obstacles when moving, and (b) the motion strategy is robust to insertion and failure of individual robots.

1.2 Organization

As is introduced in Section 1.1, this thesis focuses on introducing a decentralized coordination strategy for a large group of robots under various constraints. The rest of this thesis is organized as follows.

Chapter 2 discusses several existing approaches to multi-robot navigation. In particular, in Section 2.1 we discuss multi-robot path planning approaches that explicitly solve for the entire paths for robots. In Section 2.2, we introduce several examples of behavior-based coordination, which utilizes local interactions between individual robots to maintain safety and connectivity, and control the entire team as an entity. In Section 2.3, we discuss several recent control barrier function-based approaches that directly addresses the forward reachability of the system.

In Chapter 3, we introduce necessary background for the methodology discussed in this paper. In Section 3.1, we introduce basic notation to represent a multi-robot system and our assumptions on the robot model and environment. Section 3.2 introduces the definition of an abstract simplicial complex, especially the Vietoris–Rips complex which plays an important role in the methodology introduced in this thesis.

Chapter 4 presents the skeleton of our coordination strategy to navigate a large team of robots. To make it easier to understand, we present the *centralized* version of our strategy, and leave the decentralized framework for Chapter 5. Chapter 4 focuses on the underlying theory and theoretical proof of our strategy. We prove that the system is guaranteed to be safe and maintain connectivity while coordinating.

In Chapter 5, we present a decentralized framework for our coordination strategy. We divide our strategy into several sub-problems (stages) discussed in Chapter 4, and develope an asynchronous decentralized algorithm for each of the stages. We show correctness of the decentralized algorithms.

Chapter 6 shows simulation results of the proposed strategy navigating a large team of robots in cluttered environment. Quantitative data such as computational time, number of messages, and number of steps to achieve the goal are compared and discussed. In Chapter 7 we present conclusions.

Chapter 2

Related Work

There has been extensive work on navigating multiple robots from an initial to a final region. The approaches discussed in literature can be roughly classified into three group: planning-based coordination, behavior-based coordination, and barrier function-based coordination. In this chapter, we will discuss the advantages and limitations of the three groups of approaches respectively.

2.1 Planning-based coordination

There are a wide range of planning based approaches which seek to navigate multiple robots from start configurations to goal configurations. Typically, these approaches explicitly give a path for each of the robots from pre-defined starting configurations to a goal configuration.

Planning based approaches can be generally divided into coupled planning [3, 7, 14, 15, 19] and decoupled planning [5, 6, 10, 20]. A coupled planning approach [3, 7, 14, 15, 19] plans in the high dimensional joint configuration space of a system. As coupled approaches can potentially find all possible solutions, they usually provide strong optimality guarantees. However, a downside of these approaches is that the complexity of algorithm grows exponentially as the number of robots increases.

A decoupled approach plans in one or more lower dimensional configuration spaces. For example, there is a extensive work on prioritized planning [5, 6, 10, 20]. In this approach, robots sequentially plan paths according to a prioritization function. The prioritization function is based on heuristic. In [20], the authors develop decentralized prioritized planning algorithms which are scalable to a group of hundreds of agents. Since planning is conducted in low dimensional space, these approaches exhibits decent scalability. However, there are few guarantees on quality or optimality of the solution.

To gain the benefits of both coupled and decoupled approaches, Wagner *et al.* [21, 22] develop a search approach with sub-dimensional expansion. The search is initially done in the configuration space of an individual robot. Then collisions are detected and local dimensionality of the search space is increased to find a feasible plan. The approach is proven to be complete and optimal.

The planning-based approaches are either centralized or require a star-shaped communication graph, which is not realistic for large groups of robots.

2.2 Behavior-based Coordination

There is a group of behavior-based coordination approaches which uses swarm behavior such as consensus flocking, formation control or shape control to maintain safety and connectivity, and controls the multi-robot system as an entity. For example, For example, repulsion when robots get close to one another is designed to prevent robots from colliding, attraction when robots get far from one another is designed to prevent robots from being disconnected, while a gradient-based controller is typically used to drive the robots to a goal region. Other work seeks to preserve these properties by letting robots move in a certain formation. Among these works, quasi lattice formation, where the robots maintain almost the same distance away from each other, is widely used, because the safety and connectivity are implicitly considered in this formation. Decentralized controllers are proposed in [1, 13] to maintain a quasi lattice formation. But most prior work only presents results for obstacle-free environments or environments with sparse point-like obstacles, because maintaining a formation is very challenging in cluttered environments. However, since the control output of a behavior controller is finite, there is no safety or connectivity guarantee for the system. Moreover, in the presence of obstacles, merging these conflicting constraints into a single framework could easily lead to deadlock when guiding a large number of robots through a narrow corridor as mentioned in [2].

In order to reduce the dimensionality of path planning for multi-robot systems, [4] introduced abstraction of groups of robots using shapes such as bounding box and concentration ellipsoid, and proposed a controller to translate, rotate and deform the shape of groups of robots. However, its assumptions on the robots' distribution and the ability of shapes to deform indicates that robots are considered as particles without physical bodies. Therefore it fails to consider real world issues of safety in terms of preventing robot collisions and maintenance of connectivity.

2.3 Barrier-Based Coordination

There is another group of approaches [17, 18, 23, 24, 25] which uses control barrier functions to provide guarantees on collision-free behavior [24], connectivity maintenance [17] or both [23, 25]. Some of the approaches use an auxiliary barrier-based controller which is additive to the original controller [17, 25], while other approaches use optimization based controllers, and view barrier certificates as a constraint [23, 24]. In both ways, the system is proved to be collision free and/or preserve connectivity if it is initialized to be so, since the barrier function directly ensures the forward reachability set of the system lies in the desired region of the state space.

One downside of these barrier-based coordination methods is potential deadlock that may happen when different constraints or goals conflict with each other. For example, the goal configuration is not reached in [23] because of the conflict between goal and constraints. Few of these approaches show results in a complex cluttered environment.

Chapter 3

Background

3.1 Multi-Robot Systems

Consider a multi-robot system consisting of N homogeneous robots in a cluttered space $W \subseteq \mathbb{R}^d$, where $d \in \{2, 3\}$. Each robot has its own unique identifier (UID). For simplicity of exposition and without loss of generality, we assume that the robot UIDs are $i \in \{1, 2, \ldots, N\}$. For each robot $i \in \{1, 2, \ldots, N\}$, the position of the robot is given by $q_i \in W$. We consider the task of moving the robots from an initial position to a region centered at a goal position G.

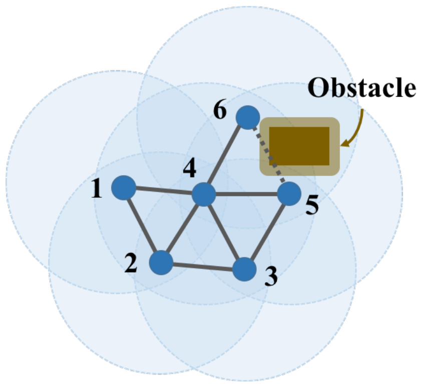


Figure 3.1: Communication graph and sensing graph of six robots with an obstacle (brown). Note that the edges in the communication graph (both solid and dashed grey segments) are not the same as edges in sensing graph (solid grey segments only), since all six robots can communicate with each other but two of them cannot see each other due to the obstacle.

Each robot is assumed to be equipped with an omni-directional range sensor which can identify whether a point within its sensing radius R_s is occupied or not. Each robot can also interact with other robots within its spatial proximity through local communication and sensing. The communication graph and sensing graph of the multi-robot system are defined as $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ and $\mathcal{G}_s = (\mathcal{V}, \mathcal{E}_s)$ respectively, where each node in $\mathcal{V}$ represents a robot. Two robots i and j can communicate if and only if the distance between them is less than or equal to the communication radius R_c (i.e. $\|q_i - q_j\| \leq R_c \Leftrightarrow (i, j) \in \mathcal{E}_c$). Two robots i and j can sense each other (i.e.

$(i, j) \in \mathcal{E}_s$) if and only if (1) the distance between them must be less than or equal to the sensing radius R_s and (2) there is no objects (e.g. obstacles and robots) on the line between robot i to robot j. We assume that $R_c = R_s = R$. It is easy to see that (a) both the communication graph and sensing graph are undirected, and (b) the sensing graph is a subgraph of the communication graph, due to the possible presence of obstacles that prevent robots from seeing some of their neighbors (see Figure 3.1).

3.2 Abstract Simplicial Complex

Before discussing the methodologies, we will introduce a basic concept in algebraic topology, namely the abstract simplicial complex, which plays an essential role in the approach developed in this thesis.

3.2.1 Abstract Simplicial Complex

Definition 1 *(Munkres, 1984, [11]) An abstract simplicial is a collection $\mathcal{S}$ of finite nonempty sets, such that if A is an element of $\mathcal{S}$, so is every nonempty subset of A.*

For example, the collection $\mathcal{K} = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}\}$ is an abstract simplicial complex. Note that different from simplicial complex which has geometric meaning, abstract simplicial complex is a pure combinatorial description.

Definition 2 *(Munkres, 1984, [11]) An element A of $\mathcal{S}$ is called a simplex of $\mathcal{S}$; its* dimension *is one less than the number of its elements. Each nonempty subset of A is called a face of A. The dimension of $\mathcal{S}$ is the largest dimension of one of its simplices.*

Therefore, $\{1, 2, 3\}$ is a 2 dimensional simplex, or 2-simplex, of $\mathcal{K}$. $\{1\}, \{2\}, \{3\}$, etc. are faces of $\{1, 2, 3\}$. The abstract simplicial complex $\mathcal{K}$ has a dimension of 2 because {1,2,3} has the largest dimension among all the simplices in $\mathcal{K}$.

Definition 3 *(Munkres, 1984, [11]) If L is a subcollection of $\mathcal{S}$ that contains all faces of its elements, then L is an abstract simplicial complex in its own right; it is called a subcomplex of $\mathcal{S}$. The d-skeleton of an abstract simplicial complex $\mathcal{S}$ consisting of all of the faces of $\mathcal{S}$ that have dimension at most d and is denoted by $\mathcal{S}^{(d)}$.*

Consider the example above, $\{\{1\}, \{2\}, \{1, 2\}\}$ is a subcomplex of $\mathcal{K}$ but not a skeleton. $\{\{1\}, \{2\}, \{3\}\}$ is the 0-skeleton of $\mathcal{K}$, which is also called the vertex set of $\mathcal{K}$.

3.2.2 Vietoris–Rips Complex

Abstract simplicial complexes give us a clean representation of the topological structure of a set of elements, or in our case, robots. However, we still don't know how to construct an abstract simplicial complex from a particular configuration of robots. In this section, we will introduce the Vietoris–Rips complex, an abstract simplicial complex that can be directly constructed from the communication graph of the robots.

Definition 4 *(Hausmann, 1995, [9]) The Vietoris–Rips complex is an abstract simplicial complex defined from a metric space. Given a set of points in a metric space and distance δ, the Vietoris–Rips complex is the collection of every finite set of points such that the distance between any two points in the set is at most δ.*

One can easily see that the Vietoris–Rips complex is an abstract simplicial complex. Assume that A is a set of points in the Vietoris–Rips complex $\mathcal{R}$. This means that the distance between any two points in A is no greater than δ. Then, for any subset $A' \subseteq A$, any two points in A' has a distance no greater than δ. Therefore, $A' \in \mathcal{R}$.

The Vietoris–Rips complex represents the topology of the communication graph for the robots. A k-simplex in a Vietoris-Rips complex $\mathcal{R}$ is formed by a subset of $(k+1)$ robots where each pair of robots in the subset has distance of at most R, i.e. the communication and sensing radius. We denote the set of all k-simplices as $\mathcal{R}_k$. It is easy to see that $\mathcal{R}_0 = \mathcal{R}^{(0)}$, and $\mathcal{R}_k = \mathcal{R}^{(k)} \setminus \mathcal{R}^{(k-1)}$ for $k \geq 1$.

For a d dimensional space, we care about simplices with dimension no greater than d. Therefore, we are particularly interested in the d-skeleton of the Vietoris–Rips complex, $\mathcal{R}^{(d)}$. For the sake of simplicity, in the rest of this thesis, we will use $\mathcal{R}$ instead of $\mathcal{R}^{(d)}$, which only contains simplicies of dimension at most d. Therefore, $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_d$.

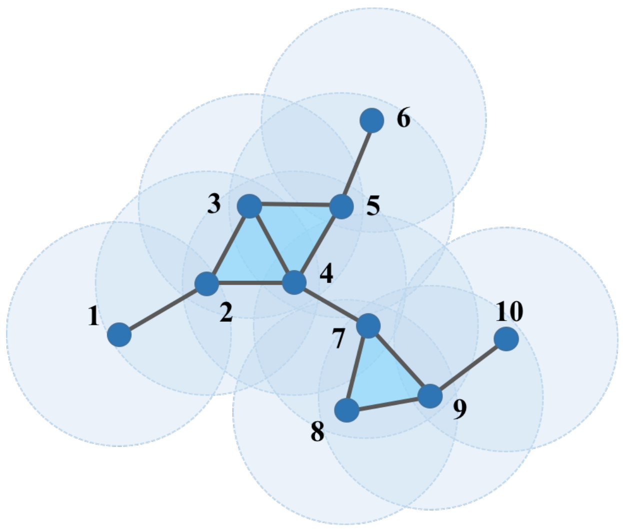


Figure 3.2: Vietoris-Rips complex formed by 10 robots in 2 dimensional space. 0-simplices are shown as blue points. Gray lines and blue triangles show 1-simplices and 2-simplices, respectively.

Specifically, a 0-simplex is each individual robot; a 1-simplex is constructed between two robots if the distance between them is less than the communication radius R; a 2-simplex is formed by a triple of robots that can all communicate with each other. In case of 3 dimensional space, we will further construct 3-simplices for quadruples of robots in the same manner. For example, in Figure 3.2, 0 -simplices (blue points) are formed by each individual robot. 1-simplices (gray lines) are constructed between two robots if the distance between them is less than the communication radius R, which is the same as edges in the communication graph. 2-simplices (blue triangles) are formed by a triple of robots that can all communicate with each other, for example, $\{2, 3, 4\}$, $\{3, 4, 5\}$ and $\{7, 8, 9\}$.

One advantage of using Vietoris–Rips complex to represent the robot team is that boundary information can be easily obtained through local computation and communication. The boundary of the robot team is represented as the frontier set, which is formed by the set of all fence simplices.

Definition 5 *In d dimensional space, a $(d-1)$-simplex formed by set of points $\mathcal{X}$ is a fence simplex if all the point x' that form a d-simplex with $\mathcal{X}$ are on the same side of the hyperplane defined by $\mathcal{X}$. A frontier set $\mathcal{F}$ is the set consisting of all fence simplices.*

For example. in Figure 3.2, $\{7, 8\}$ is a fence 1-simplex because there is only one point, 9, that forms a 2-simplex with $\{7, 8\}$, while $\{3, 4\}$ is not a fence 1-simplex because $\{2, 3, 4\}$ and $\{3, 4, 5\}$ are all 2-simplices, and robots 2 and 5 are on different sides of $\{3, 4\}$.

As a generalization of frontier set, we further define k-degenerate frontier set as all the k-simplices that are on the boundary of the Vietoris–Rips complex.

Definition 6 *A k-degenerate frontier set $\mathcal{F}^k$ is the set consisting of all k-simplices, with $0 \leq k \leq d-2$, such that it is a fence simplex when all the points are projected onto a $k+1$ dimensional space.*

Chapter 4

Methodology

Inspired by [16], in our approach, only a subset of robots moves in each step. At each step, a new desirable unoccupied position, called the *frontier node* (see Section 4.1) and *tail robot* (see Section 4.2) for the robot team is chosen. Subsequently, a path of robots from the tail robot to the frontier node is found in the sensing graph. Then, a "push" action is performed along the path, i.e. each robot, starting with the robot closest to the frontier node moving into its position, moves one position to occupy the forward position along this path that was vacated by the previous robot moving along the path. By moving in this pattern, most of the communication and sensing graph do not change each step except for the part associated with the frontier node and tail robot. An illustration of this motion pattern is shown in Figure 4.1. We assume that both the communication graph and sensing graph are connected in the initial configuration.

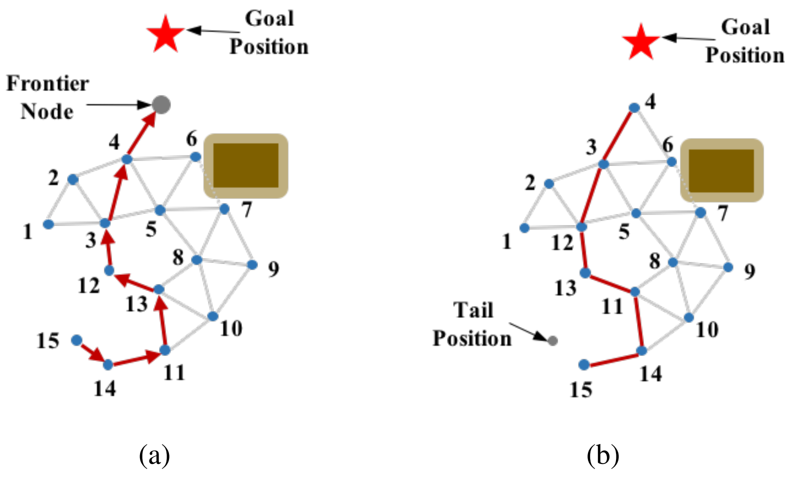


Figure 4.1: Motion pattern of the robots over sensing graph (grey edges). (a) Frontier node (grey) and tail robot (robot 15) are chosen. The shortest path from tail robot to frontier node is $15 \rightarrow 14 \rightarrow 11 \rightarrow 13 \rightarrow 12 \rightarrow 3 \rightarrow 4 \rightarrow$ Frontier Node. (b) Each robot moves to occupy the forward position that was vacated by the previous robot moving along the path, with robot 4 moving to the frontier node.

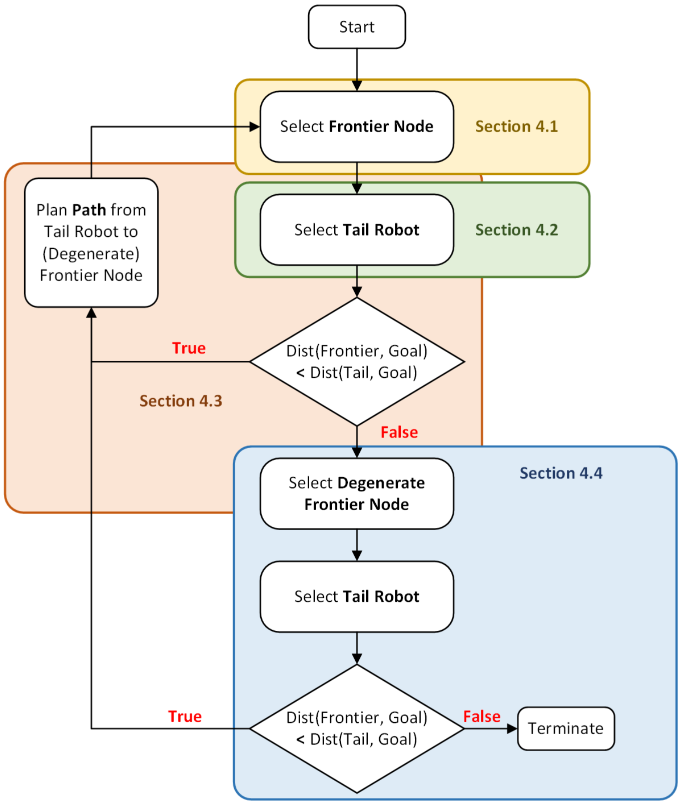

Start
Select **Frontier Node**
Section 4.1
Select **Tail Robot**
Section 4.2
Plan **Path** from Tail Robot to (Degenerate) Frontier Node
True
Dist(Frontier, Goal) < Dist(Tail, Goal)
Section 4.3
False
Section 4.4
Select **Degenerate Frontier Node**
Select **Tail Robot**
True
Dist(Frontier, Goal) < Dist(Tail, Goal)
False
Terminate

Figure 4.2: Illustration of centralized framework of our algorithm and organization of this section

In our approach, each robot i has a stepwise goal position denoted as g_i, which can be the position of the frontier note, forward position along its path or same as the robot's current position. We assume that there is a low level controller (e.g. PD controller) that can drive the robot to the stepwise goal position during that step.

We note that since we select the frontier node, tail robot and plan path using an updated communication and sensing graph at each step, our algorithm is inherently robust to failure and insertion of individual robots, as well as change of goal position.

There are three key components in our approach: frontier node selection, tail robot selection and path planning. For the rest of this section, we will introduce our criterion and method for these three components. We will also prove that with our approach, the robot team can stay connected and navigate without collision in a static environment.

4.1 Frontier Node Selection

As is discussed at the beginning of this chapter, the frontier node is important since it determines the direction of motion of the robot team. A wise choice of frontier node can also give the system desirable properties, such as formation, connectivity, etc. In this section, we introduce our approach to select the frontier node.

For a given fence, we define a *virtual node* as a node that is located in the outer side of the

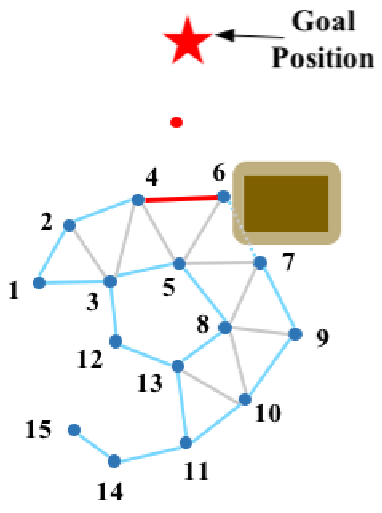


Figure 4.3: Fence simplices (cyan) and selected frontier node (red dot) for a given configuration. The corresponding fence simplex $\{4, 6\}$ is also shown in red.

fence and its distance to all the robots that form the fence is $R - \delta$, where δ is a tunable positive number. The frontier node is the virtual node that has the shortest Euclidean distance to the goal position. We use this greedy strategy as we assume that the robots do not have knowledge about the environment. If the robots have knowledge about the environment, then we can use a more advanced heuristic instead of Euclidean distance can be used, for example, length of shortest path in 2d space with obstacles.

One remark is that, in the frontier set, there may be fence simplices that are part of an internal hole. Since the selection criterion for the best fence, is smallest distance from the goal, a virtual node corresponding to the internal fence simplex can never be chosen as the frontier node.

Definition 7 *The extended sensing graph is $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$, where $\mathcal{V}_e$ is the union of nodes in sensing graph $\mathcal{V}$ and frontier node f, $\mathcal{E}_e$ is the union of edges in sensing graph $\mathcal{E}_s$ and (f, v) for any node v in the best fence simplex F_{min} such that f is visible to v.*

4.2 Tail Robot Selection

After the frontier node has been selected, our algorithm selects a robot as the tail of the path to the frontier node. As is illustrated in Figure 4.1, the edges associated with the tail robot will be removed for the next time step. Therefore, naive tail robot selection criteria that are based only on the distance to goal do not preserve the connectivity of the robots. Figure 4.4 illustrates an example that such naive criteria to select the tail robot cause the robots to become disconnected.

In order to preserve connectivity throughout coordination, we select the tail robot based on the *spanning trees* of the extended sensing graph $\mathcal{G}_e$. We note that deleting a leaf robot from the spanning tree does not make the original graph disconnected. Based on the idea that the deepest node in a tree is always a leaf, we select the robot that is deepest in the spanning tree as tail robot.

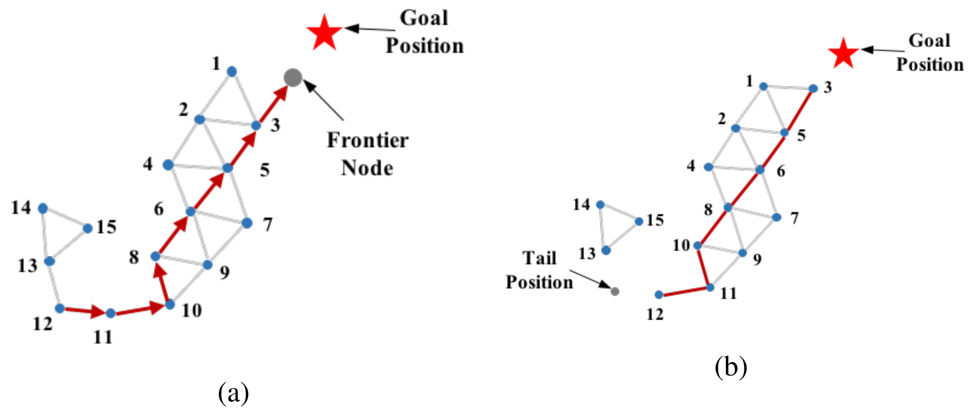


Figure 4.4: A case where naive criteria for tail robot selection fails to preserve connectivity. (a) The shortest path from a naively selected tail robot (robot 12) to the frontier node. (b) Selecting robot 12 as tail robot causes the robots to become disconnected at the next time step.

We can show that such selection of tail robot will not make the sensing and communication graph disconnected. Another reason that we use a spanning tree to find the tail robot is that it can be easily implemented in a decentralized manner, and the structure of the spanning tree enables fast interchange of messages inside the robot team, which will be discussed in detail in Chapter 5.

Theorem 1 (Connectivity Guarantees) *If the communication and sensing graph are connected in the initial configuration, then the communication and sensing graph will remain connected while moving.*

Proof: Consider the spanning tree, T for the extended sensing graph $\mathcal{G}_e = (\mathcal{V}, \mathcal{E}_e)$. We define the graph $\mathcal{G}'_e$ induced by $\mathcal{G}_e$ deleting a leaf node v of spanning tree T and its associated edges. It is easy to see that $\mathcal{G}'_e$ is a spanning subgraph of the sensing graph for the next step. Therefore, if $\mathcal{G}'_e$ is connected, then both the sensing graph and the communication graph are connected for the next step. By induction, we can prove that if initially connected, $\mathcal{G}_c$ and $\mathcal{G}_s$ will stay connected while moving. We prove that the graph $\mathcal{G}'_e$ is connected by constructing a tree T' which is proved to be a spanning tree of $\mathcal{G}'_e$. Construct T' by deleting v and its associated edge (there is only one such edge) from the spanning tree T. Since v is a leaf of T, T' is a connected tree. According to the definition of the spanning tree, T contains vertices $\mathcal{V}_e$ and edges $\mathcal{E}_T \subseteq \mathcal{E}_e$, and T is connected. Therefore, T' contains vertices $\mathcal{V}_e \setminus \{v\}$ and edges $\mathcal{E}_T \setminus \{(v, n) : n \in \mathcal{V}\}$. Therefore, T' is a spanning tree of $\mathcal{G}'_e$.

Remark 1 *Note that there may exist more than one spanning tree of the extended sensing graph $\mathcal{G}_e$. The selection of spanning tree is an interesting design choice since different spanning trees may result in slightly different behavior of the system. We will introduce the particular spanning tree we use in Section 5.3.*

4.3 Path Planning

After the frontier node and the tail robot is selected. We can plan a path of robot such that the "push" action can be made. For a desired path, we need 1) for each robot on the path, it is able to move directly to the current location of its parent in the path without causing collision, and 2) the number of robots moving at each time step should be as small as possible. Therefore, we can use the shortest path in the extended sensing as the path of robot. We can prove that the resulting action is collision free.

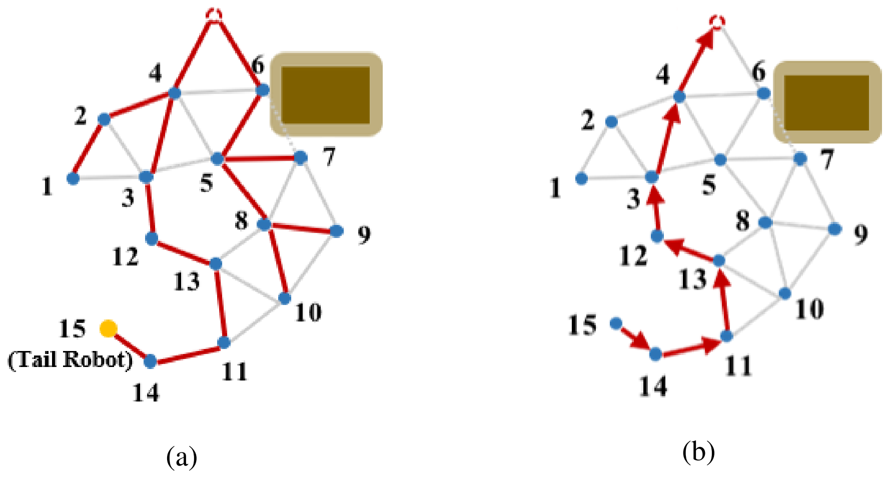


Figure 4.5: (a) Illustration of tail robot selection using spanning tree. Robot 15 (yellow) is selected since it is the deepest leaf in the spanning tree. (b) The actual path (red arrows) planned by our algorithm.

Theorem 2 (Collision Free Guarantees) *We assume that a robot can only sense a point if and only if there is no obstacle or robot sitting on the path between it and a certain point within its sensing radius. If the obstacles are all static, then our algorithm can guarantee that the robots will not collide with other robots or obstacles during locomotion if all the obstacles are far enough from the initial region.*

Proof: Remember that only a subset of robots which forms a path in the extended sensing graph $\mathcal{G}_e$ is selected to move at each time step. Therefore, (1) the path from the robot to its master is clear, and (2) the step-wise goal position is either being vacated by its master (for the robots not moving to the frontier node), or is free throughout the step (for the robot moving to the frontier node). We only need to show that there is no path intersection between robots. When the robots have formed the lattice band formation, there is no intersection because there is no intersecting edge in the sensing graph. Before the robots have formed the lattice formation, we can assume that the sensing graph and communication graph is the same since the obstacles are far from the initial region. In this case, based on a fact from geometry, there always exists a path with no intersection which is shorter than the path with intersecting edges. Therefore, our algorithm will never plan a path with intersecting edges. The movement does not result in any inter-robot collision or collision with obstacles.

Remark 2 *Note that if the tail robot is closer to the goal than the frontier node, however, the robots will not move along the path because it will cause the robot to move further away from the goal. This makes our algorithm terminate implicitly and achieve a rendezvous-like behavior near the goal.*

4.4 Handling Corridors

Even though the lattice formation can handle most of the situations in cluttered spaces, there may be cases where a corridor is so narrow that the lattice can not be fully expand, and therefore there will be no frontier fence to be expanded (for example, Figure 4.6). In that case, we have to identify whether the robots are stuck at the opening of a narrow corridor. This can be done by using the idea of a degenerate fence discussed in Section 3.2.2 when the frontier node is further from goal than the tail node.

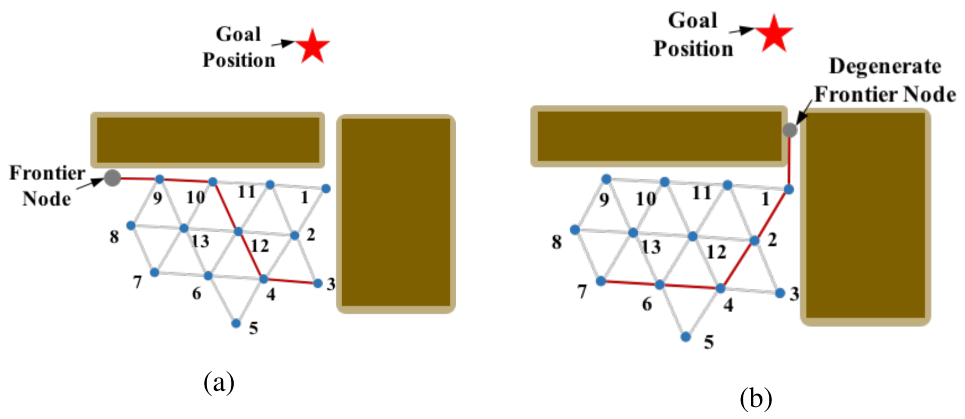


Figure 4.6: Handling narrow corridors which the robot can not pass with lattice formation. (a) The frontier node is selected as the virtual node expanded by $\{8, 9\}$. The tail robot is selected as robot 3. However, the tail robot is closer to the goal than the frontier node. (b) Our algorithm moves on the expand the degenerate frontier, and drives the robot through the corridor.

In Figure 4.6a, for example, the frontier node is selected as the virtual node expanded by $\{8, 9\}$. The tail robot is selected as robot 3. However, robot 3 is closer to the goal than the frontier node. So our algorithm moves on to expand a degenerate frontier. For each degenerate fence, we choose a virtual node that is nearest to the goal as is shown in Figure 4.6b, and select the degenerate frontier and its corresponding degenerate frontier node with the same procedure as described in Section 5.2. Then, the path planning stage is the same as in previous sections. If the tail robot is closer to the goal than the degenerate frontier node, however, the robots will not move along the path because it will cause the robot to move further away from the goal. This makes our algorithm terminate implicitly and achieve a rendezvous-like behavior near the goal.

Chapter 5

A Decentralized Framework

In the previous chapter, we have discussed the centralized version of our approach. In this chapter, we will introduce a decentralized framework to implement our approach which provides scalability and robustness. Our framework consists of several sub-algorithms, including construction of the Vietoris–Rips complex, selection of the frontier node, selection of the tail robot and planning of the path.

For each step, each of the sub-algorithms (stages) is run in a decentralized and asynchronous way, and ends implicitly when there is no message sent regarding that particular stage. Therefore, we introduce a pre-defined time limit for each of the stage s as $T_{L,s}$. Each robot measures how long there has been no message received during a stage s. If the time exceeds $T_{L,s}$, the robot will decide the current stage is over, and start to send messages for the next stage $s+1$. When a robot at stage s receives a message belonging to stage $s+1$, it switches its current stage to $s+1$, and starts the procedure for the new stage. Since each of the sub-algorithm is asynchronous, any of the robots can initiate the next stage, and each of the sub-algorithms can converge to the correct results. Therefore, each of our sub-algorithms is decentralized and no global information is needed when switching between stages. For the rest of this chapter, we will introduce each of the sub-algorithms.

5.1 Decentralized Construction of Simplicial Complex

Algorithm 1 describes how each robot calculates its local simplicial complex (only the simplicial complexes that include itself) in a decentralized manner. This makes our work different from [16], which calculates the simplicial complex for the multi-robot system in a centralized way. As discussed in Section 3.2.2, $\mathcal{R}_k$ denotes the set of all k-simplices.

At the beginning, all the set of simplices is initialized to the empty set, except $\mathcal{R}_0 = \{u\}$ (line 2). Then, the asynchronous procedure is initiated by each robot sending a message to all of its direct neighbors (line 4). The parameters for SENDMSG$(i, u, \ldots)$ are defined as follows, i is the target of the message, u is the source of the message, and others are the content of the message. Similar is RECVMSG(), but there is no target in this function.

After that, the algorithm calculates the simplicial complex of each robot i and its direct neighbors (line 6-28). First, when a message is received from u's direct neighbor u' for the first time (line 9-11), the UID of u' is added to the local 0-simplices list, and the the pair of robot u and u' are added to 1-simplices list. Then robot u sends a message to its neighbor with the updated local simplicial complex.

Algorithm 1 Decentralized Local Simplicial Complex Construction Procedure

```
 1: procedure DecentralizedSimplicialComplex($u$,$\mathcal{N}_u$)
 2:     $\mathcal{R}_0 \leftarrow \{u\}$.$\mathcal{R}_1 \leftarrow \emptyset, \ldots, \mathcal{R}_d \leftarrow \emptyset$
 3:     for all $i \in \mathcal{N}_u$ do
 4:         SendMsg($i$,$u$,$\mathcal{R}_0$,...,$\mathcal{R}_d$)
 5:     end for
 6:     while $\{u', \mathcal{R}'_0, \ldots, \mathcal{R}'_d\} \leftarrow$RecvMsg() do
 7:         // Construct local 0 and 1 simplices
 8:         if $\{u, u'\} \notin \mathcal{R}_1$ then
 9:             $\mathcal{R}_0 \leftarrow \mathcal{R}_0 \cup \{u'\}$
10:             $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{u, u'\}$
11:             SendMsg($i$,$u$,$\mathcal{R}_1$,...,$\mathcal{R}_d$)
12:         end if
13:         // Construct local 2 to $d$-simplices
14:         $\mathcal{S} \leftarrow \mathcal{R}_0 \cap \mathcal{R}'_0$
15:         for $k = 2, \ldots, d$ do
16:             for all $\mathcal{S}_k \subseteq \mathcal{S}$ with $|\mathcal{S}_k| = k + 1$ do
17:                 $\mathcal{C}_{k-1} \leftarrow \{S \subset \mathcal{S}_k : |S| = k\}$
18:                 if $\mathcal{C}_{k-1} \subseteq \mathcal{R}_{k-1}$ then
19:                     $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \mathcal{S}_k$
20:                 end if
21:             end for
22:         end for
23:         if $\mathcal{R}_0$,...,$\mathcal{R}_d$ is updated then
24:             for all $i \in \mathcal{N}_u$ do
25:                 SendMsg($i$,$u$,$\mathcal{R}_0$,...,$\mathcal{R}_d$)
26:             end for
27:         end if
28:     end while
29: end procedure
```

Then, the algorithm constructs the local 2-simplices (and 3-simplices in 3 dimensional cases). The algorithm starts by calculating the union of sets of 0-simplices for robots u, and u', which is the set of their common neighbors denoted as $\mathcal{S}$ (line 14). For k range from 2 up to d, for each subset of $\mathcal{S}$ consisting of k robots $\mathcal{S}_k$, the algorithm checks whether all k subset of robots $\mathcal{C}_{k-1}$ in $\mathcal{S}_k$ is within $\mathcal{R}_{k-1}$ (line 16-21). If so, it means that $\mathcal{S}_k$ can form a k-simplicial complex, since the distance between each pair of robots is within communication radius. For example, if $\{i, l\}$ and $\{j, l\}$ are in the 1-simplices list for robot i, and robot j, respectively, then $\{i, j, l\}$ should be a 2-simplex in $\mathcal{R}_2$ of both robot i, j, and l. If the simplicial complex is updated when processing the message, then the robot will send a message to all of its neighbors (line 23-27).

We note that our algorithm works in an asynchronous way, that is, our method only requires pairwise communication between robots, and the order of robots receiving messages does not matter. Our algorithm will end implicitly when each robot u constructs all the simplices that contain the robot u.

5.2 Decentralized Selection of the Frontier Node

To select the frontier node in a decentralized manner, each robot first calculates its local frontier set $\mathcal{F}_u$, i.e., the set of all fence simplices that include the robot. By definition of a fence simplex, a $(d-1)$-simplex is a fence simplex if either there is no d-simplex that has that $(d-1)$-simplex as a boundary, or all the robots that are in d-simplex but not in the $(d-1)$-simplex lie in the same side of the $(d-1)$-simplex hyperplane. The robot iterates all the local $(d-1)$-simplices and selects the virtual node closest to the goal. Selecting a fence candidate can be done by each robot without any communication.

Algorithm 2 Decentralized Frontier Selection Procedure

```
1:  procedure DecentralizedFrontier($u$,$\mathcal{R}_{d-1}$,$\mathcal{R}_d$,$\mathcal{N}_u$,$G$)
2:      $\mathcal{F}_u \leftarrow \mathcal{R}_{d-1}, \mathcal{P}_u \leftarrow \emptyset$
3:      for all $r_{d-1} \in \mathcal{R}_{d-1}$ do
4:          $R_d \leftarrow \{r : \{r_{d-1}, r\} \in \mathcal{R}_d\}$
5:          if OnBothSide($r_{d-1}$,$R_d$) then
6:              $\mathcal{F}_u \leftarrow \mathcal{F}_u \setminus r_{d-1}$
7:          else
8:              $p \leftarrow$ExtendFrontier($r_{d-1}$, $R_d$)
9:              $\mathcal{P}_u \leftarrow \mathcal{P}_u \cup p$
10:         end if
11:     end for
12:     $d_u^* \leftarrow \min_{p\in\mathcal{P}_u}\{\|G-p\|\}$
13:     $p_u^* \leftarrow \arg\min_{p\in\mathcal{P}_u}\{\|G-p\|\}$
14:     $f_u^* \leftarrow \arg\min_{f\in\mathcal{F}_u}\{\|G - \text{ExtendFrontier}(f, R_d)\|\}$
15:     for all $i \in \mathcal{N}_u$ do
16:         SendMsg($i$,$u$,$d_u^*$,$p_u^*$,$f_u^*$)
17:     end for
18:     while $\{u', d_{u'}^*, p_{u'}^*, f_{u'}^*\} \leftarrow$RecvMsg() do
19:         if $d_{u'}^* < d_u^*$ then
20:             $d_u^* \leftarrow d_{u'}^*, p_u^* \leftarrow p_{u'}^*, f_u^* \leftarrow f_{u'}^*$
21:         end if
22:     end while
23: end procedure
```

After selecting the local frontier set, each robot extends the fence simplices and calculates the location of virtual nodes $p \in \mathcal{P}_u$. Some fences would not be expanded for a variety of reasons: (a) the existence of obstacles on the estimated position of a virtual node, (b) the virtual node position is not visible to the robot. For all the fence simplices that can be expanded, the position of virtual node p and its distance to the goal d is calculated.

Then, the frontier node f^* and its corresponding fence simplex F^*, is selected in a decentralized way. The robots initialize its belief of frontier node f_u^* as the local virtual node nearest to the goal. When a robot u receives a message from its neighbor u' containing its belief of frontier node $f_{u'}^*$, and its distance to the goal $d_{u'}^*$, it determines whether the neighbor's belief is closer to the goal than its own (u's) belief. If so, robot u updates its belief and sends a message with this update to its direct neighbors. Finally, every robot will reach a consensus about the frontier node, i.e., $f_u^* = f^*$ for all u.

Remark 3 *Since the procedure is very much the same for selecting the degenerate frontier node which is used to handle narrow corridors, we will not introduce the decentralized procedure for that in this chapter.*

5.3 Decentralized Tail Robot Selection

Algorithm 3 Decentralized Tail Robot Selection

```
 1: procedure DecentralizedTail($u$,$q_u$,$\mathcal{N}_u$,$f$,$G$)
 2:     if $u \in F^* \wedge (q_u, f) \in \mathcal{E}_e'$ then
 3:         $h \leftarrow 1, m \leftarrow 0$
 4:     else
 5:         $h \leftarrow \infty, m \leftarrow i$
 6:     end if
 7:     for all $i \in \mathcal{N}_u \wedge (q_u, q_i) \in \mathcal{E}_e$ do
 8:         SendMsg($i$,$u$,$h$)
 9:     end for
10:     // Construct a spanning tree rooted at the frontier node
11:     while $\{u', h'\} \leftarrow$RecvMsg() do
12:         if $h > h' + 1$ then
13:             $h \leftarrow h' + 1, m \leftarrow u'$
14:             for all $i \in \mathcal{N}_u \wedge (q_u, q_i) \in \mathcal{E}_e$ do
15:                 SendMsg($i$,$u$,$h$)
16:             end for
17:         end if
18:     end while
19:     // Find the leaf robot that is deepest in the tree
20:     $tid \leftarrow u, th \leftarrow h, td \leftarrow \|G - q_u\|$
21:     for all $i \in \mathcal{N}_u$ do
22:         SendMsg($i$,$u$,$t_{id}$,$t_{hop}$,$t_{dist}$)
23:     end for
24:     while $\{u', t'_{id}, t'_{hop}, t'_{dist}\} \leftarrow$RecvMsg() do
25:         if $t_{hop} < t'_{hop}$ then
26:             $t_{id} \leftarrow t'_{id}$, $t_{hop} \leftarrow t'_{hop}$, $t_{dist} \leftarrow t'_{dist}$
27:             for all $i \in \mathcal{N}_u$ do
28:                 SendMsg($i$,$u$,$t_{id}$,$t_{hop}$,$t_{dist}$)
29:             end for
30:         else if $t_{hop} = t'_{hop} \wedge t_{id} \neq t'_{id}$ then
31:             if $t_{dist} < t'_{dist}$ then
32:                 $t_{id} \leftarrow t'_{id}$, $t_{dist} \leftarrow t'_{dist}$
33:                 for all $i \in \mathcal{N}_u$ do
34:                     SendMsg($i$,$u$,$t_{id}$,$t_{hop}$,$t_{dist}$)
35:                 end for
36:             end if
37:         end if
38:     end while
39: end procedure
```

After the frontier node has been selected, our algorithm selects a robot as the tail of the path to the frontier node. Our algorithm presented in Algorithm 3 uses a decentralized way to select a tail robot such that (a) its deletion does not harm the connectivity of the graph, and (b) the constructed path with the frontier node as head has minimal length.

First, our algorithm implicitly constructs a hop-optimal spanning tree [12] (line 3-18). Algorithm 3 is initiated by one or several robots in the fence simplex corresponding to the frontier node, denoted as F^*, that can sense the frontier node f^* (in other words, robots v suh that (v, f^*) is an edge in the extended sensing graph $\mathcal{E}_e$). The spanning tree provides two advantages: (1) since removal of any leaf of the spanning tree does not harm the connectivity of the graph (proven in Section 4.2), we can choose a leaf in the spanning tree as the tail robot, (2) due to the hop-optimality of the spanning tree, a shortest path from the tail robot to the frontier node is exactly the path from the tail robot (leaf) to the frontier node in the spanning tree.

Another important remark about our procedure for constructing the spanning tree is that each robot does not know the structure of the spanning tree. It is only aware of its master m (parent) in the spanning tree, and the number of hops (h) between it and the root.

Then, based on the idea that the deepest node in a tree is always a leaf, the algorithm starts finding the robot that has the deepest hop (line 20-38). At the beginning, each robot believes itself to be a leaf, with tail robot id $t_{id} = u$, tail robot hop number $t_{hop} = h$, tail robot distance to goal position $t_{dist} = \|G - q_u\|$. If the robot receives a message indicating that (1) another robot is deeper than its current belief of tail robot (line 25) or (2) another robot is equally deep as its current belief but it is further from the goal (line 31), it will update its belief, i.e., it will not consider itself as a leaf anymore and send a message to its neighbors.

5.4 Plan Path from Tail to Frontier Node

At the beginning of this sub-algorithm, the stepwise goal positions for robots are initialized as the robots' current positions. If the tail robot is closer to the goal than the frontier node, then the robot will not move since this movement will drive the robot team further away from the goal. Then we need to check the degenerate frontier node described in Section 4.4, and select a new tail robot corresponding to the degenerate frontier node.

Algorithm 4 Decentralized Plan Path to Frontier Procedure

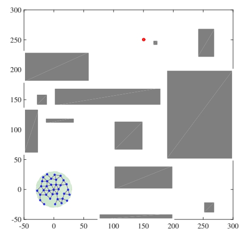
```
 1: procedure DecentralizedPathPlan($u$,$q_u$,$\mathcal{N}_u$)
 2:     $g_u \leftarrow q_u$
 3:     if $u = t_{id}$ then
 4:         $g_u \leftarrow q_m$
 5:         if $m \neq 0$ then
 6:             SendMsg($m$,$u$)
 7:         end if
 8:     end if
 9:     while $\{u'\}$ ←RecvMsg() do
10:         $g_u \leftarrow q_m$
11:         if $m \neq 0$ then
12:             SendMsg($m$,$u$)
13:         end if
14:     end while
15: end procedure
```

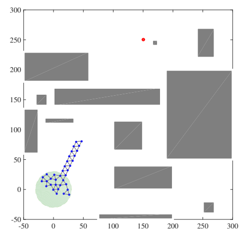
As discussed in Section 5.3, the shortest path from the tail robot to the frontier node is exactly the path in the hop-optimal spanning tree. Therefore, the tail robot will first set its goal position for this time step to be the position of its current (spanning tree) master (line 4) and send a message to its master. When a robot receives a message, it means that it is on the path, and should move to its master's position in this time step. It will then send a message to its own master until the robot actually occupies the frontier node (head of the path). In this way, each robot will move towards its stepwise goal position for this time step (see Section 4.1).

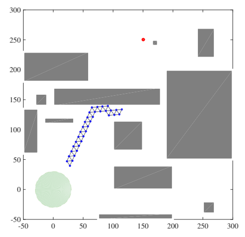
Chapter 6

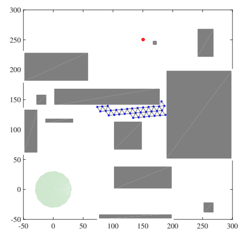
Simulation Results

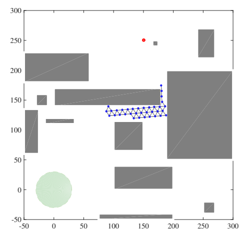
In this section, we illustrate the performance of our algorithm by simulations in 2 dimensional scenarios. In the simulation, the communication radius and sensing radius are $R = 10$. Each robot has circular body with radius $r = 1$. In initial configuration, the robots are arbitrarily placed in a region centered by a starting position $(0, 0)$. The objective of the robots is to move to a goal region centered at $(150, 250)$.

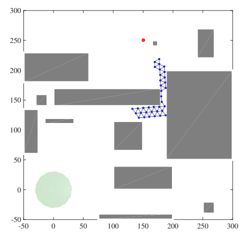

(a) step 1

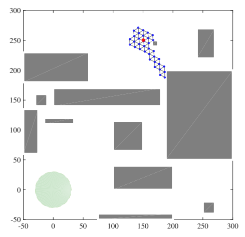

(b) step 16


(c) step 40


(d) step 64


(e) step 67


(f) step 79


(g) step 105

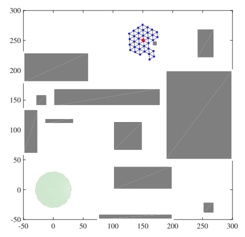
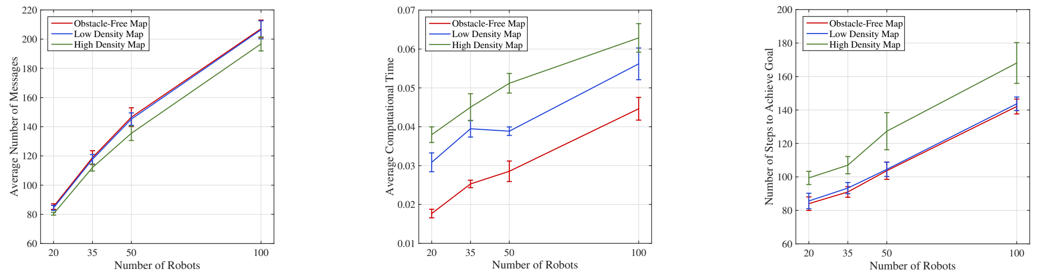

(h) step 115

Figure 6.1: Snapshots of a group of 35 robots (blue circles) moving from initial configuration (step 1) toward the goal position (red circle). The robots implicitly move in a lattice band formation, pass though corridors, and rendezvous around goal position (step 115).

Figure 6.1 illustrates our algorithm in a challenging scenario with 35 robots. A large portion of the area is occupied by obstacles and there is also a narrow corridor that does not allow robots to pass in a lattice band formation. Figure 6.1a shows the initial configuration of the robots, where robots are placed near the starting point. Please note that the robots are not necessarily in a lattice formation initially. Then the robots start moving to a lattice band formation by extending frontier fences (Figure 6.1b). As is shown in Figure 6.1c, the lattice band deforms when encountering obstacles. Then the robots gather at a corner before passing through the corridor (Figure 6.1d). After that, robots start passing through the corridor by extending the degenerate frontier fence, i.e., the robots move in a broken line formation when passing through the corridor (Figure 6.1e). After passing the corridor, the robots autonomously re-form the lattice band. Subsequently, the robots start gathering near the goal position (Figure 6.1g). Figure 6.1h shows the final configuration of the robots. The robots succeed in achieving a rendezvous-like behavior in a finite number of steps, even with the existence of an obstacle near the goal position.

To show the scalability of our algorithm, we tested our algorithm with different numbers of robots ranging from 20, 35, 50 to 100. We also tested our algorithm with 50 robots on 3 different maps: (a) obstacle-free map, (b) low density map with relatively small portion (10%) of area occupied by obstacles, and (c) high density map with relatively high portion (30%) of area occupied (this map is shown in Figure 6.1). All cases are tested under the same starting and goal positions as is shown in the map. For each of the settings, there are 10 trials with different initial configurations near the same starting point. The average computational time in seconds, average number of messages for each robot per step, and number of steps to reach goal region is shown in Figure 6.2. All the three properties grows linearly as the number of robots increases. Moreover, map density does not have significant influence on average number of messages. However, the average computational time increases by a constant as the map density increases. One possible reason is that the computational time for collision check for obstacles increases as the number of robots increases. This also matches the constant increase and the invariance of message numbers. Number of steps to converge to goal configuration also increases by a constant as the map becomes denser. This occurs because the distance that the robots have to travel to the goal increases as the map becomes denser.



(a) (b) (c)

Figure 6.2: Quantitative results for a group of 20, 35, 50 and 100 robots in three 2-dimendional maps with different density level (10 trials each). The average number of messages, average computational time and number of steps to achieve goal all scale almost linearly. Map density variance causes approximately constant difference for computational time and number of steps as the number of robots grows, but does not have significant influence on the number of messages.

Chapter 7

Conclusion

This thesis presents a novel decentralized behavior-based coordination strategy to navigate a large team of robot through a cluttered environment under various constraints, including avoiding inter-robot collision, preserving connectivity, and avoiding collision with static obstacles. Under this coordination strategy, the robot team implicitly forms a deformable lattice formation and navigates toward a given goal location with incremental motion. Key performance with respect to the average number of messages, average computational time and average number of time steps to converge on different sized robot teams in maps with different obstacle density levels were demonstrated in simulations to validate the effectiveness and scalability of the proposed algorithm. The algorithm has multiple advantageous properties:

1. The algorithm scales linearly in the number of robots as is shown in simulation results.
2. The algorithm enables robot teams to deform flexibly considering the incrementally sensed obstacles in the environment.
3. The algorithm guarantees collision avoidance and connectivity preservation without suffering from deadlock.
4. The algorithm is robust to insertion or failure of individual robots since it utilizes incremental movement to coordinate the robot team.

Bibliography

[1] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. Flocking for multi-robot systems via the null-space-based behavioral control. *Swarm Intelligence*, 4(1):37–56, 2010. 2.2

[2] Nora Ayanian and Vijay Kumar. Abstractions and controllers for groups of robots in environments with obstacles. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3537–3542. IEEE, 2010. 2.2

[3] Jerome Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 1991. 2.1

[4] Calin Belta and Vijay Kumar. Abstraction and control for groups of robots. *IEEE Transactions on robotics*, 20(5):865–875, 2004. 2.2

[5] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1419–1424. IEEE, 1986. 2.1

[6] Carlo Ferrari, Enrico Pagello, Jun Ota, and Tamio Arai. Multirobot motion coordination in space and time. *Robotics and autonomous systems*, 25(3-4):219–229, 1998. 2.1

[7] Yi Guo and Lynne E Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 2612–2619. IEEE, 2002. 2.1

[8] Allen Hatcher. *Algebraic topology.* Cambridge university press, 2002. 1.1

[9] Jean-Claude Hausmann et al. On the vietoris-rips complexes and a cohomology theory for metric spaces. *Ann. Math. Studies*, 138:175–188, 1995. 4

[10] Wenhao Luo, Nilanjan Chakraborty, and Katia Sycara. Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints. In *2016 American Control Conference (ACC)*, pages 148–154. IEEE, 2016. 2.1

[11] James R Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984. 1, 2, 3

[12] Sasanka Nagavalli, Andrew Lybarger, Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Aligning coordinate frames in multi-robot systems with relative sensing information. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 388–395. IEEE, 2014. 5.3

[13] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006. 2.2

[14] David Parsons and John Canny. A motion planner for multiple mobile robots. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 8–13. IEEE, 1990. 2.1

[15] Jufeng Peng and Srinivas Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research*, 24(4):295–310, 2005. 2.1

[16] Rattanachai Ramaithitima, Michael Whitzer, Subhrajit Bhattacharya, and Vijay Kumar. Sensor coverage robot swarms using local sensing without metric information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3408–3415. IEEE, 2015. 1.1, 4, 5.1

[17] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12):1411–1423, 2013. 2.3

[18] Lorenzo Sabattini, Cristian Secchi, Nikhil Chopra, and Andrea Gasparri. Distributed control of multirobot systems with global connectivity maintenance. *IEEE Transactions on Robotics*, 29(5):1326–1332, 2013. 2.3

[19] Jacob T Schwartz and Micha Sharir. On the piano movers' problem: Iii. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2(3):46–75, 1983. 2.1

[20] Prasanna Velagapudi, Katia Sycara, and Paul Scerri. Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4603–4609. IEEE, 2010. 2.1

[21] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3260–3267. IEEE, 2011. 2.1

[22] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015. 2.1

[23] Li Wang, Aaron D Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2659–2664. IEEE, 2016. 2.3

[24] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 2017. 2.3

[25] Michael M Zavlanos, Ali Jadbabaie, and George J Pappas. Flocking while preserving network connectivity. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2919–2924. IEEE, 2007. 2.3