Decentralized Coordinated Motion for a Large Team of Robots Preserving Connectivity and Avoiding Collisions

Anqi Li, Wenhao Luo, Sasanka Nagavalli, *Student Member, IEEE*, Katia Sycara, *Fellow, IEEE*

Abstract— **We consider the general problem of moving a large number of networked robots toward a goal position through a cluttered environment while preserving network communication connectivity and avoiding both inter-robot collisions and collision with obstacles. 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 distance, 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.**

I. Introduction

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 [1], exploration of unknown environments [2] and environmental sampling [3]. In real-world applications, how to guide the large-scale robot team towards goal regions faces several challenges. First, the system should be safe, in that the robots should not collide with one another or obstacles in the environment. Second, the robots should remain connected while coordinating. Third, the system should be scalable as the number of robots grows.

We study the problem of the coordinated 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.

This research has been sponsored in part by ONR Grant N0001409-10680, an NSERC PGS D scholarship and a Siebel Scholarship.

*The authors are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {anqil1, wenhaol, snagaval}@andrew.cmu.edu, katia@cs.cmu.edu.

There has been extensive work on navigating multiple robots from an initial to a final region, such as multi-robot path planning [4]–[7] and swarm control [8]–[13]. In multi-robot path planning approaches, the entire paths for each individual robot connecting explicitly pre-defined starting configurations to goal configurations are computed either in high-dimensional joint configuration space [4], [5] or in a decentralized manner [6], [7]. However, planning-based approaches usually either scales poorly to large-scale robot teams [1], or required star-shaped communication graph [6], [7], which is not realistic for large groups of robots.

Swarm control focuses on robots' distributed incremental movements governed by predefined control laws towards the goal region such as flocking strategy [8]–[10] or formation control [11]. To avoid inter-robot collisions and maintain connectivity, auxiliary controllers that handle these constraints are usually combined into frameworks of swarm control. For example, bio-inspired strategies embed virtual repulsive force to achieve collision free movements while preserving inter-robot connectivity via attraction forces [10]. Other work seeks to apply control laws on algebraic connectivity in order to maintain global connectivity [14], [15]. However, 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 [13].

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 simplicial complex from algebraic topology [17], we reduce our problem dimension by formally describing the communication graph of the robot team as simplicial complex that provides feasible frontier nodes for computing robots' incremental motions. The resulting lattice-formation motion pattern in free space is similar with [8], [9] where the inter-robot collision and connectivity are implicitly considered, but note that we are not explicitly specifying the parameters of the rendezvous-like lattice formation as done in [8], [9]. 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 3. 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 during moving, and (b) the motion strategy is robust to insertion and failure of individual robots.

II. Background

A. Basic Notations

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.

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_{sen} 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 1).

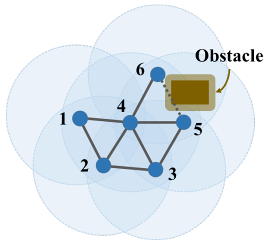


Fig. 1: Communication graph and sensing graph of six robots with an obstacle (brown). Note that the edges in the communication graph (both solid and dashed grey segments) are not the same as edges in sensing graph (solid grey segments only), since all six robots can communicate with each other but two of them cannot see each other due to the obstacle.

B. Vietoris-Rips complex

The Vietoris-Rips simplicial complex [17] 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 $(k+1)$ points where each pair of points 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 $\mathcal{R}_k$. For a d dimensional space, we will construct up to d-simplices. Therefore, $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_d$. 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.

Definition 1: A $(d-1)$-simplex formed by set of points $\mathcal{X}$ is a *fence simplex* in d dimensional space if all the point x' that form a d-simplex with $\mathcal{X}$ are on the same side of the hyperplane defined by $\mathcal{X}$. For example. in Figure 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 $\{3, 4\}$.

Definition 2: A *frontier candidates subcomplex* $\mathcal{F}$ is the subcomplex consisting of all fence $(d-1)$-simplices.

Definition 3: A *degenerate frontier candidates subcomplex* $\mathcal{F}^*$ is the subcomplex 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.

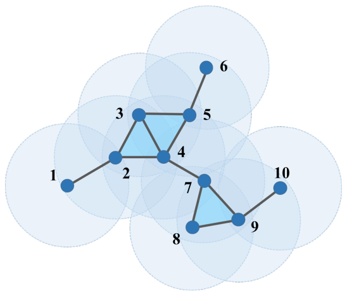


Fig. 2: Vietoris-Rips complex formed by 10 robots in 2 dimensional space. 0 -simplices (blue points) are formed by each individual robot. 1-simplices (gray lines) are constructed between two robots if the distance between them is less than the communication radius R, which is the same as edges in the communication graph. 2-simplices (blue triangles) are formed by a triple of robots that can all communicate with each other

III. Approach Overview

We assume that both the communication graph and sensing graph are connected in the initial configuration. Inspired by [16], in our approach, only a subset of robots moves in each step. At each time step, a new desirable unoccupied position, called the *frontier node* (see Section III-B) and *tail robot* (see Section III-C) for the robot team is chosen. Subsequently, a shortest path from the tail robot to the frontier node is found in a decentralized manner (see Section III-D). 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 3.

Note that since we select the frontier node, tail robot and plan path using changed communication and sensing graph for each time step, our algorithm is inherently robust to failure and insertion of individual robot, as well as change of goal position.

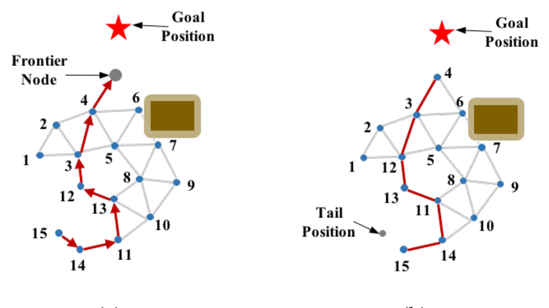


Fig. 3: Motion pattern of the robots over sensing graph (grey edge). (a) Frontier node (grey) and tail robot (robot 15) are chosen. The shortest path from tail robot to frontier node is $15 \rightarrow 14 \rightarrow 11 \rightarrow 13 \rightarrow 12 \rightarrow 3 \rightarrow 4 \rightarrow$ Frontier Node. (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.

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

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 to 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 have 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 message for the next stage $s+1$. When a robot at stage s receives a message belonging to the stage $s+1$, it switches its current stage to $s+1$, and start 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, this provides scalability.

A. 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 II-B, $\mathcal{R}_k$ denotes the set of all k-simplices.

At the beginning, all the set for 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$(u, \ldots)$, but there is no parameter for the target.

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.

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-simplices 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 do not matter. Our algorithm will end implicitly when each robot u constructs all the simplices that contain the robot u.

Algorithm 1 Decentralized Local Simplicial Complex Construction Procedure

```
 1: procedure DECENTRALIZEDSIMPLICIALCOMPLEX($u$,$\mathcal{N}_u$)
 2:     $\mathcal{R}_0 \leftarrow \{u\}$.$\mathcal{R}_1 \leftarrow \emptyset$, ..., $\mathcal{R}_d \leftarrow \emptyset$
 3:     for all $i \in \mathcal{N}_u$ do
 4:         SENDMSG($i$,$u$,$\mathcal{R}_0$,...,$\mathcal{R}_d$)
 5:     end for
 6:     while $\{u', \mathcal{R}'_0, \ldots, \mathcal{R}'_d\}$ ←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
```

B. Decentralized Selection of the Frontier Node

For a given fence, we define a *virtual node* as a node that is located in the outer side of the fence and its distance to all the robots that form the fence is $R - \delta$, where δ is a tunable positive number. We choose the fence based on (1) whether its corresponding virtual node is visible to a least one of the robots forming the fence, and (2) the distance between the virtual node candidate and the goal. We call the procedure of calculating the virtual node corresponding to a fence candidate as *extending the fence* candidate. We will describe the procedure of selecting the frontier node, namely the most desirable node that will be the head of the path that the robots will follow.

First, each robot calculates its local fence candidates, i.e., the fence $(d-1)$-simplices that include the robot. By definition of a fence, a $(d-1)$-simplex is a fence candidate 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. Selecting a fence candidate can be done by each robot without any communication. The robot iterates all the $(d-1)$-simplices and selects the fence whose distance from the goal is smallest. In the fence candidate list, there may be fences that are part of the fence set of an internal hole. Since the selection criterion for the best fence, the frontier fence, is smallest distance from the goal, a fence that is part of a border of an internal hole will not be selected as a fence to be expanded.

After selecting local fences, each robot extends it by calculating their corresponding virtual nodes. The virtual node and the selected $(d-1)$-simplex fence form a d-simplex. Since the robots are desired to move in a lattice formation, the position of the virtual node is calculated to be equidistant from each of the robots in the fence (see Figure 3). Some fences may not be able to 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 for robot, that is, robot can not go straight into that position because of the existence of obstacles on its way. For all the fences that can be expanded, the position of virtual node and its distance to the goal is calculated. Then, the robots initialize its belief of best fence to be the one whose corresponding virtual node is nearest to the goal.

Then, the fence with virtual node nearest to the goal, the frontier fence f_{min}, is selected in a decentralized way. That is, when a robot u receives a message from its neighbor u' containing its belief of frontier fence, its corresponding frontier node, and distance from the frontier node to the goal, it calculates whether the frontier fence and node in u''s message is closer to the goal than its own (u's) best fence and virtual node, 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 best fence, the frontier fence and corresponding frontier node for this step.

The final selected frontier node is the "head" of the path that the robots will be following.

Definition 4: 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.

C. Decentralized 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 3, the edges associated with the tail robot will be removed for 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 illustrates an example that such naive criteria to select tail robot cause the robots to become disconnected. However, directly identifying whether removing an edge can cause the graph to be disconnected requires centralized information so it is not appropriate for our decentralized framework.

Algorithm 2 Decentralized Tail Robot Selection

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

Our algorithm presented in Algorithm 2 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.

Therefore we develop an algorithm that implicitly constructs a hop-optimal spanning tree [18] (line 3-19). Algorithm 2 is initiated by one or several robots in the best fence F_{min} 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 IV), 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.

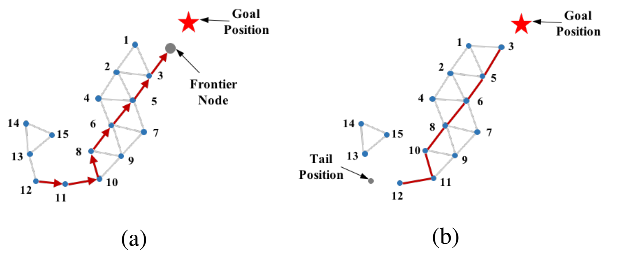


Fig. 4: A case where naive criteria for tail robot selection fails to preserve connectivity. (a) The shortest path from a naively selected tail robot (robot 12) to the frontier node. (b) Selecting robot 12 as tail robot causes the robots to become disconnected at the next time step.

Another important remark about our procedure of 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 22-40). 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 message indicating that (1) another robot is deeper than its current belief of tail robot (line 27) or (2) another robot is equally deep as its current belief but it is further from the goal (line 33), it will update its belief, i.e., it will not consider itself as leaf anymore, and send a message to its neighbors.

D. 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. After the path from the tail robot to the frontier node has been generated (as described in the previous sections), if the tail robot is further away from the goal than the frontier node, the tail robot will first set its goal position for this time step to be the position of its current (spanning tree) master 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 3).

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.

E. 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 5). 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 II-B when the frontier node is further from goal than the tail node.

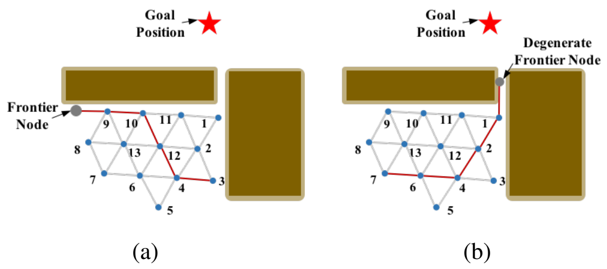


Fig. 5: Handling corridors. (a) The frontier node is selected as the virtual node expanded by $\{8, 9\}$. The tail robot is selected as robot 3. However, the tail robot is closer to the goal than the frontier node. (b) Our algorithm moves on the expand the degeneralized frontier, and drives the robot through the corridor.

In Figure 5a, 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 5b, and select the degenerate frontier and its corresponding degenerate frontier node with the same procedure as described in Section III-B. Then, the path planning stage is the same as in previous sections.

IV. Proof of Guarantees

Theorem 1: (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.

Proof: 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). Therefore, the movement does not result in any inter-robot collision or collision with obstacles. ■

Theorem 2: (Connectivity Guarantees) If the communication and sensing graph are connected in the initial configuration, then the communication and sensing graph will remain connected throughout moving.

Proof: Consider the spanning tree, T, which we constructed implicitly in Algorithm 2 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 is connected for the next step. By induction, we can prove that if initially connected, $\mathcal{G}_c$ and $\mathcal{G}_s$ will stay connected during 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 $\mathcal{G}_e'$. ■

V. 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 $(150, 250)$.

Figure 6 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 6a 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 6b). As is shown in Figure 6c, the lattice band deforms when encountering obstacles. Then the robots gather at a corner before passing through the corridor (Figure 6d). 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 6e). After passing the corridor, the robots autonomously re-form the lattice band. Subsequently, the robots start gathering near the goal position (Figure 6g). Figure 6h 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). 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 7. 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.

VI. Conclusion

In this paper we present a decentralized algorithm for moving large teams of robots through a cluttered environment. The algorithm has multiple advantageous properties: (a) it scales linearly in the number of robots, (b) it enables robot teams to deform flexibly considering the incrementally sensed obstacles in the environment, (c) it achieves simultaneously collision avoidance and connectivity preservation, and (d) it is robust to insertion or failure of individual robots. Key performance with respect to the average number of messages, average computational time and average number of time steps to converge on different scale of robot teams in maps with different obstacle density levels were demonstrated in simulations to validate the effectiveness and scalability of the proposed algorithm.

References

[1] H. Wang, A. Kolling, N. Brooks, S. Owens, S. Abedin, P. Scerri, P.-j. Lee, S.-Y. Chien, M. Lewis, and K. Sycara, "Scalable target detection for large robot teams," in *Proceedings of the 6th international conference on Human-robot interaction*. ACM, 2011, pp. 363–370.

[2] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3254–3259.

[3] W. Luo , S. S. Khatib, S. Nagavalli, N. Chakraborty, and K. Sycara, "Distributed knowledge leader selection for multi-robot environmental sampling under bandwidth constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[4] S. Swaminathan, M. Phillips, and M. Likhachev, "Planning for multi-agent teams with leader switching," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5403–5410.

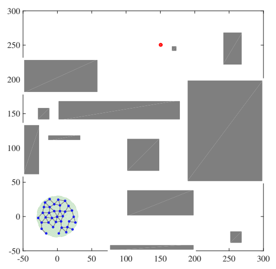
[5] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[6] W. Luo, N. Chakraborty, and K. Sycara, "Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 148–154.

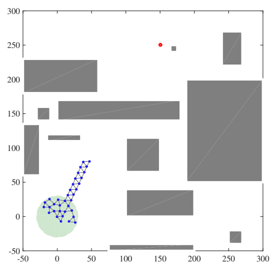
[7] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[8] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

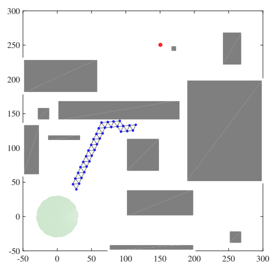
[9] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Flocking for multi-robot systems via the null-space-based behavioral control," *Swarm Intelligence*, vol. 4, no. 1, pp. 37–56, 2010.



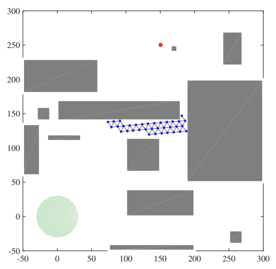
(a) step 1



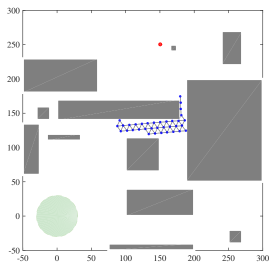
(b) step 16



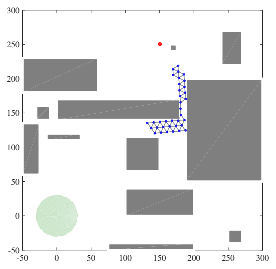
(c) step 40



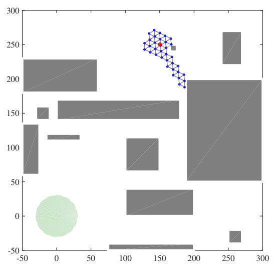
(d) step 64



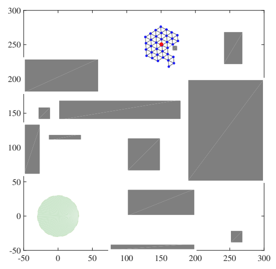
(e) step 67



(f) step 79

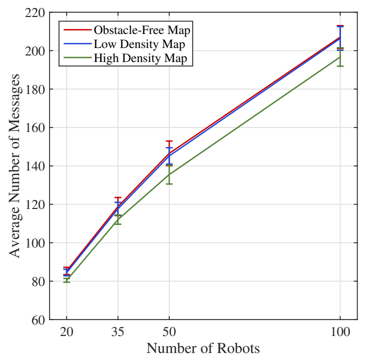


(g) step 105

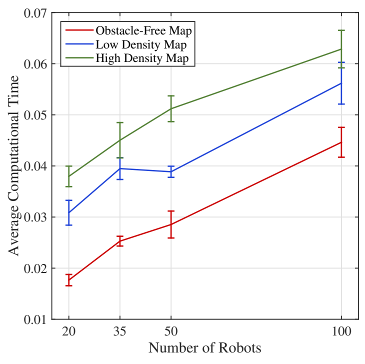


(h) step 115

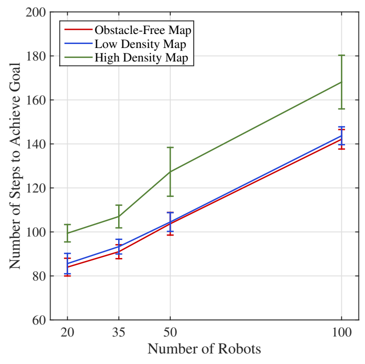
Fig. 6: Snapshots of a group of 35 robots (blue circles) moving from initial configuration (step 1) toward the goal position (red circle). The robots implicitly move in a lattice band formation, pass though corridors, and achieve a rendezvous behavior around goal position (step 115).



(a)



(b)



(c)

Fig. 7: Simulation 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 average number of messages.

[10] M. M. Zavlanos, A. Jadbabaie, and G. J. Pappas, "Flocking while preserving network connectivity," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 2919–2924.

[11] X. Yan, J. Chen, and D. Sun, "Multilevel-based topology design and shape control of robot swarms," *Automatica*, vol. 48, no. 12, pp. 3122–3127, 2012.

[12] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on robotics*, vol. 20, no. 5, pp. 865–875, 2004.

[13] N. Ayanian and V. Kumar, "Abstractions and controllers for groups of robots in environments with obstacles," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3537–3542.

[14] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[15] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[16] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar, "Sensor coverage robot swarms using local sensing without metric information," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3408–3415.

[17] A. Hatcher, *Algebraic topology.* Cambridge university press, 2002.

[18] S. Nagavalli, A. Lybarger, L. Luo, N. Chakraborty, and K. Sycara, "Aligning coordinate frames in multi-robot systems with relative sensing information," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 388–395.