Decentralized Connectivity Maintenance for Multi-Robot Systems Under Motion and Sensing Uncertainties using a Partially Observable Markov Decision Process

Bradley Collicott*, Daniel Neamati*, and Alexandros Tzikas*
Stanford University, Stanford, CA, 94305, United States

Collaborative multi-robot systems outperform unconnected robot systems in applications that benefit from cooperation and coordination. Connectivity maintenance is a prerequisite for collaboration to exchange mission-critical information among agents. However, motion and sensing uncertainties complicate the connectivity maintenance problem. Solutions that do not incorporate these uncertainties risk collisions between agents or loss of connectivity. In this work, we tackle the problem of connectivity maintenance in a multi-robot system under both motion and sensing uncertainties. We formulate the problem as a Partially Observable Markov Decision Process (POMDP) and determine a control policy for each agent in the network that ensures both connectivity maintenance and collision avoidance. We learned an offline policy that jointly prescribes the policy for the agents. Online, each agent independently executes the polity in a decentralized manner, since a centralized execution would be infeasible in the case of loss of connectivity. We simulate our algorithm in scenarios where the leader has a separate task, which we treat as a predefined trajectory. The followers then work collaboratively to maintain connectivity and avoid collisions. We demonstrate that our learned policy performs better than the random policy in at least one of the two objectives.

I. Introduction

Collaborative multi-robot systems have attracted growing interest due to their ability to coordinate and cooperate, which allows them to outperform single agents in complex tasks. Applications include search and rescue, wildfire containment, and exploration in treacherous terrain. In each case, the autonomous agents must share time-critical information with teammates or a base station to direct their mission strategy. This information could be the location of toppled buildings, the rate of a wildfire's advancements, or hazards in the area. Such scenarios naturally involve heterogeneous teams where some autonomous agents have specialized tools and lead the mission. Other agents may be devoted to connectivity maintenance with specialized hardware for high-speed data transfer. Connectivity maintenance is a prerequisite for collaboration in a multi-agent system. There is a communication path between any two robots as long as they maintain connectivity over the whole team. Stochasticities related to agent motion and measurements are inherent in all agents in any environment. This noise leads to deviations from the mathematically modeled nominal planned motion of the robot, while also corrupting its observations. Such motion deviations and measurement errors can lead to loss of connectivity, if they remain unaccounted for. Therefore, it is critical to account for motion and sensing uncertainties when dealing with connectivity maintenance in a multi-robot system.

A. Prior Work

Several works have tackled the problem of connectivity maintenance analytically, exploring decentralized algorithms that have attractive scalability and efficient communication properties. For example, [1, 2] derive a decentralized gradient-based controller for each agent in order to maintain the connectivity of the system above a pre-specified threshold. However, they fail to account for the robots' motion and sensing uncertainties. The state-of-the-art analytical method in [3] accounts for motion and sensing uncertainties. In this paper, the authors represent the system as a weighted graph, where each node represents a robot and each edge represents the communication connection between the two robots it is connected to. The edge weights are a function of line-of-sight communication and collision avoidance parameters that take into consideration the existing uncertainties, due to the underlying noise in the motion and sensing of the agents. The connectivity of the system is determined by the algebraic connectivity parameter that indicates

*Ph.D. Student, Aeronautics and Astronautics.

whether there exists a communication path (direct or indirect) between all agents. The authors then implement a decentralized gradient-based controller that provides control inputs to maintain the algebraic connectivity above a threshold, while avoiding collisions. Although the framework achieves its goal, in Eq. (12) of [3], the authors select specific functions to account for the uncertainty in their definition of the edge weights of the weighted graph. These functions are hyperparameters of the system and their selection can greatly impact performance. Finally, a general issue with hand-crafted controllers is their limited universality and their specific form greatly impacting performance.

Learning-based methods are receiving traction, given that they provide end-to-end solutions that can map observations directly to control policies. In contrast to analytical methods like [3], learning-based methods do not require an explicit functional form and learn this policy directly from the problem formulation or data. In [4], the authors propose a model-free reinforcement learning (RL) approach with behavior cloning to enable navigation of a multi-robot system in an environment with unknown objects, while maintaining connectivity. Connectivity maintenance is posed as a constraint, and a decentralized execution of the shared policy follows its centralized learning, which uses the experiences of all the agents. However [4], does not account for the partial observability of a multi-robot system, since it assumes that all agents can observe all other agents. In [5], the geometric center of a multi-robot team must efficiently reach a way-point without collisions in an unknown environment, while maintaining connectivity. The authors propose a deep RL-based method that derives end-to-end policies for the multi-robot system. To guarantee connectivity, a constraint satisfying parametric function (CSPF) is introduced to represent the policy. The CSPF is composed of a neural network and an optimization module to constrain the policy, leading to connectivity guarantees. Based on the observation tuple of all agents, the robot team determines the action tuple during the centralized training phase. In execution, the policy is divided into the components corresponding to each agent, leading to a decentralized execution. In [5], it is also assumed that each robot observes every other robot. In addition, connectivity is not determined via the algebraic connectivity, but rather by the distance between agents, restricting the structure of the system, since in reality communication could exist indirectly between agents further away from each other by passing messages through other agents in the network. Finally, the CSPF must be pre-specified and the selection is not unique, leading to variations in performance. In [6], a deep actor-critic RL approach is considered for the same problem as [5], based on a POMDP formulation. A policy is found that maps raw LiDAR data to control signals, following a centralized-training-decentralized-execution paradigm. The same connectivity definition as in [5] is considered, although in this work connectivity is incorporated into the reward, not as an additional constraint, because more flexibility regarding loss of connectivity is assumed. Even in this case, the approach in [5] outperforms classic artificial potential fields. Finally, [7] proposes a deep Q-learning network that preserves the connectivity of the multi-agent system. However, state uncertainty is not accounted for in this work, nor is collision avoidance.

B. Contributions of this Work

In this work, we propose a learning-based approach for connectivity maintenance in a multi-robot system that is able to map agent beliefs directly to control signals. Learning-based methods are able to generalize well in unseen scenarios, while not requiring the explicit selection of many parameters. This contrasts [3] where the functions that account for the uncertainty must be chosen and [5] where the CSPF must be determined. Our method is based on a POMDP formulation to account for the underlying state and sensing uncertainty. The more natural Dec-POMDP (for decentralized systems) formulation is not used due to the computational intractability of Dec-POMDP problems [8–10]. We follow a centralized-training-decentralized-execution paradigm. A policy that maps a tuple of all agents' observations to a joint action is learned and this policy is deployed individually by each agent using their belief of the global state. This enables the application of the policy in cases where connectivity must be recovered while accounting for the partial observability of the system for each agent. Also, for simplicity in this first implementation, we represent the problem in a discretized manner similar to [11], while integrating the connectivity requirement in the reward function. This approach combines simplicity and competitive results. Though, this discrete approach may be relaxed in future work. For example, we could learn the policy in the discretized work and subsequently apply kernel smoothing to generate a policy continuous space. In order to obtain an applicable framework, we also consider collision avoidance requirements in the reward function. The algebraic connectivity metric is used to allow for further flexibility in the deployment of the multi-agent system, allowing for indirect communication paths through the system. We test our implementation in two scenarios with a leader and follower robot in which the policy is learned for the follower robot, while the leader robot has a pre-specified trajectory. This matches the case where the leaders have an external objective and rely on the followers to maintain connectivity. Our results show that our learned policy outperforms a random policy with respect to at least one of the metrics: connectivity maintenance and collision avoidance.

C. Paper Organization

The rest of the paper is organized as follows: in Section II, the general problem is formulated mathematically as a POMDP, in Section III, our proposed approach is described in detail for the leader-follower scenarios, in Section IV, our simulation results are presented and analyzed. Finally, conclusions are found in Section V, future work is mentioned in VI and the team members' contributions can be found in Section VII.

II. Problem Formulation Overview

In this proposed work, we transform the problem in [3] into a POMDP. More explicitly, recall that a POMDP is defined as a tuple $\langle \mathscr{D}, \mathcal{S}, \boldsymbol{A}, T, R, \boldsymbol{O}, O, I \rangle$. We can formulate the connectivity maintenance problem in a multi-robot system under motion and sensing as a discretized POMDP through the following conversion:

1) The set $\mathscr{D}$ of agents represents the robots. This set has cardinality $|\mathscr{D}|$.
2) The joint state space $\mathcal{S}$ of the environment is discretized onto a 2D grid. A state $\boldsymbol{s} = (s_1, ..., s_{|\mathscr{D}|})$ contains the positions of the robots ($s_i = (x_i, y_i)$).
3) The joint action space $\boldsymbol{A}$ contains all tuples $\boldsymbol{a} = \left(\boldsymbol{a_1}, \ldots, \boldsymbol{a}_{|\mathscr{D}|}\right)$, where each $\boldsymbol{a_i}$ is the action of robot i, chosen from a discrete set. For each robot i, $\boldsymbol{a_i}$ contains its control input as dictated by the applied policy.
4) The transition probability function $T : P(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$ can be calculated through the motion models of the agents. This continuous distribution is then discretized to ensure compatibility with the discretized state space $\mathcal{S}$.
5) The reward function R is a function of the algebraic connectivity λ_2 of the system and the existence of collisions with obstacles or among agents.
6) The joint observation space $\boldsymbol{O}$ contains all tuples $\left(\boldsymbol{o_1}, \ldots, \boldsymbol{o}_{|\mathscr{D}|}\right)$, where each $\boldsymbol{o_i}$ is the independent observation of the position of robot i, chosen from a discrete set.
7) The observation probability function $O : P(\boldsymbol{o} \mid \boldsymbol{s}, \boldsymbol{a})$ can be extracted by using the motion model, along with the sensing model per agent. Again, this distribution is discretized to be compatible with the overall model.
8) The initial state distribution I is a deterministic user-defined state.

III. Proposed Approach

In this section, we detail our proposed approach for a set-up involving leader and follower robots. Leader robots have a pre-determined trajectory, while follower robots take actions in order to maintain connectivity. Leader and follower robots are treated differently in the possible transitions they can complete, as discussed in a later section. The leader-follower set-up is commonly used in testing connectivity maintenance algorithms, as shown in [3].

A main consideration throughout our implementation was to ensure our code was compatible with the package POMDPs.jl [12], in order to make use of the built-in solvers. Part of the motivation for the discretized implementation is also the abundance of solvers for discrete state-action spaces in POMDPs.jl.

A. States

For our problem, we specify an $n \times n$ sized 2D grid of possible positions for each robot. The state space is of size $(n \times n)^{|\mathscr{D}|}$, where n is the side length of the square grid and $|\mathscr{D}|$ is the total number of robots, including both leaders and followers. The leader robots' trajectory is not known beforehand, so information about the leaders' positions must be incorporated in the state of the system. At any instance, the system state is a tuple of Cartesian indices with one Cartesian index per robot. For example, we may have two robots, with the leader at cell $(2, 2)$ and the follower at cell $(8, 8)$, as seen in Figure 1. Then the state is $\boldsymbol{s} = ((2, 2), (8, 8))$. We can index the state space via linear indices, as also shown in Figure 1. Moreover, the obstacles' locations are modeled as accessible states with a large negative reward. We do not include terminal states, though obstacle collisions could be considered terminal states in future work.

B. Actions

A robot can move to any adjacent cell (8 possibilities: E, NE, N, NW, W, SW, S, WE) or remain at its current cell (stay), for a total of 9 action choices at each instance. We include the leaders' actions in the action space because information about the leaders' trajectory is not known before runtime, although they are fixed at runtime. The action space of the problem consists of tuples $(a_1, ..., a_{|\mathscr{D}|})$ such that there are $9^{|\mathscr{D}|}$ possible action tuples. The joint action state and current joint action are depicted in Figure 1 for an example including one leader and one follower robot. In this case, the leader moves east and the follower elects to stay in place.

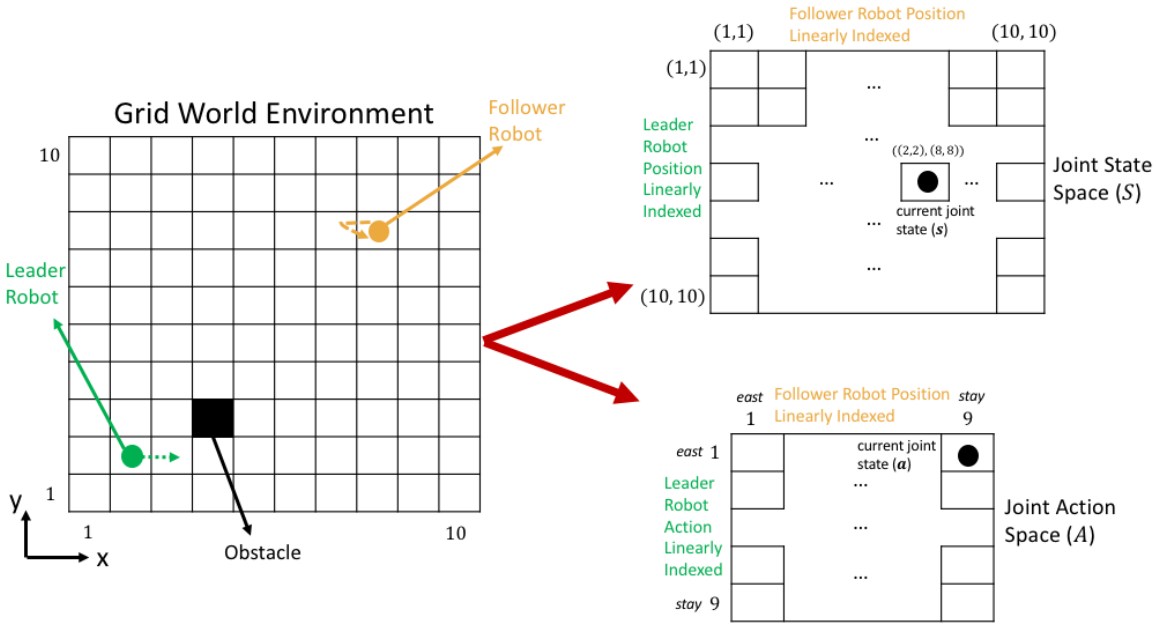


Fig. 1 Grid World Environment with Joint State and Action for a Scenario with One Leader and One Follower Robot.

C. Transitions

Agents do not move deterministically in their environment due to uncertainty in the motion model. Therefore, the transition probability $T(\boldsymbol{s'} \mid \boldsymbol{s}, \boldsymbol{a})$ for each robot is a probability distribution over the cells adjacent to its true position in our discretized environment. We assume a discrete Gaussian distribution over the adjacent positions of the robot with given bin width, b_t, and standard deviation, σ_t. If a potential transition leads the robot out of the map, then the probability associated with the out-of-bounds position is added to the probability of staying at the current position. For simplicity, we assume the leader robots deterministically transition to their desired state. The transition distribution parameters for an agent intending to transition from state s_i to state s_j are given as

$$\text{Leader:} \quad T(s_j \mid s_i, a_j) = 1.0 \tag{1}$$

$$\text{Follower:} \quad T(s_{j+k} \mid s_i, a_j) = T(s_{j+k+1} \mid s_i, a_j) = \int_{\frac{b_t}{2}(2k-1)}^{\frac{b_t}{2}(2k+1)} \mathcal{N}(x \mid 0, \sigma_t^2) dx \quad k \in 0:4, \tag{2}$$

where the adjacent positions are enumerated and the distribution is visualized in Figure 2. The transition distributions for each robot are combined to create the transition function from a state $\boldsymbol{s} = (s_1, ..., s_{|\mathscr{D}|})$ and action $\boldsymbol{a} = (a_1, ..., a_{|\mathscr{D}|})$ to a state $\boldsymbol{s'}$. The joint transition probability is computed as the product of individual robot transition probabilities assuming that transitions are independent of one another, and the resulting distribution is stored as a sparse categorical distribution over possible tuples of reachable positions. Note that the set of reachable states $(9^{|\mathscr{D}|})$ is much smaller than the full state space $\left((n \times n)^{|\mathscr{D}|}\right)$, since the agents can only transition to one of the 8 adjacent cells or stay in the current cell.

D. Observations

Each robot position s_i is mapped to an observation o_i using a probability model that is derived using a Gaussian and a uniform distribution. The robot position is observed at the true position with a probability obtained from the center of a Gaussian with given bin width, b_o, and standard deviation, σ_o. However, the position may be observed as one of the adjacent states with a uniform probability. The observation distribution parameters for a single agent are given as

$$O(s_9) = \int_{\frac{-b_o}{2}}^{\frac{b_o}{2}} \mathcal{N}(x \mid 0, \sigma_o^2) dx, \qquad O(s_i) = \frac{1 - O(s_9)}{8} \quad i \in 1:8. \tag{3}$$

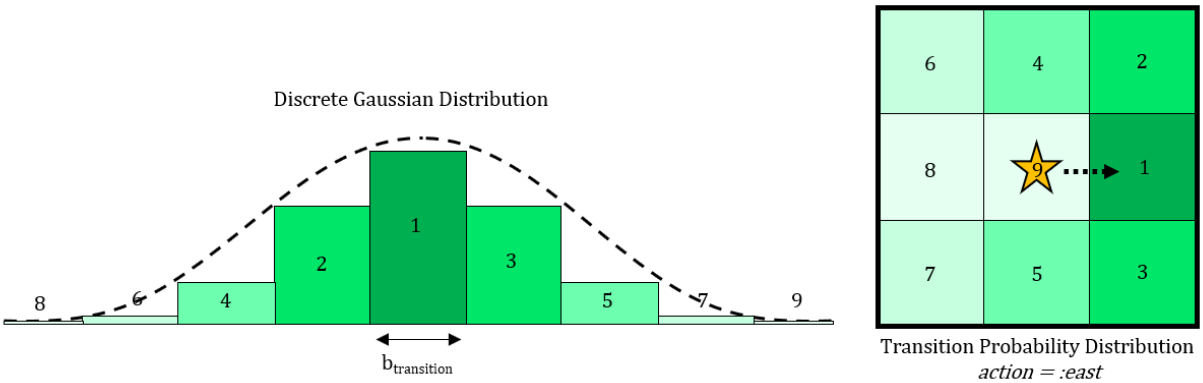


Fig. 2 Transition Probability Distribution for Follower Action.

The joint observation distribution is created as a sparse categorical distribution assuming independence between individual observations. In the decentralized policy rollout, this joint distribution is sampled by each agent to actualize the observation as a tuple of all agents' states, using its belief about the other agents' actions. This observation is used to update the state belief of each agent, which the learned policy will map to an action tuple.

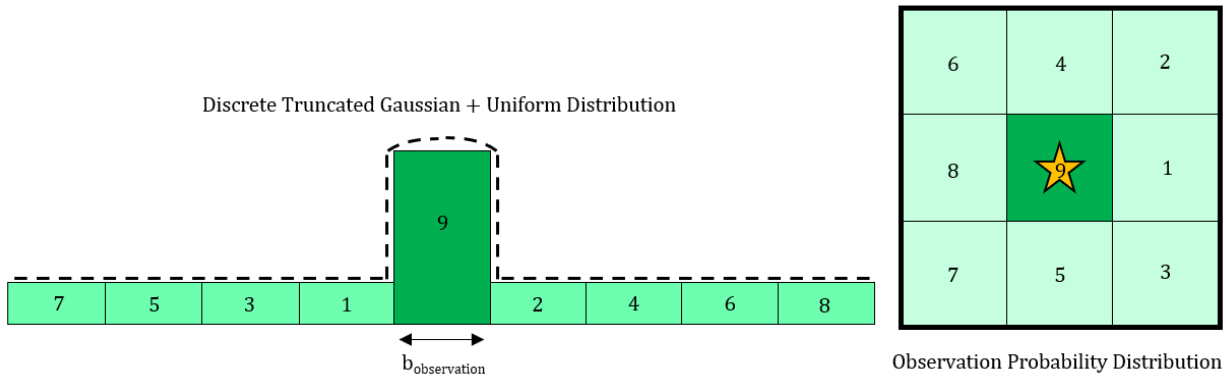


Fig. 3 Observation Probability Distribution for Each Agent.

E. Rewards

There is one common-central reward for the complete multi-robot system, dependent on the current system's state (the tuple of agents' positions) only. The reward consists of the sum of three terms defined in Eq. (4).

$$R = R(\boldsymbol{s}) = r_i^{\text{obstacle coll}}(\boldsymbol{s}) + \sum_{i \in \mathscr{D}} r_i^{\text{agent coll}}(\boldsymbol{s}) + r^{\text{connect}}(\boldsymbol{s}), \tag{4}$$

where

- $r_i^{\text{obstacle coll}}(\boldsymbol{s}),\ \forall i$, is the negative reward due to collision with obstacles. If an agent is closer to an obstacle than a predefined number of cells (obstacle collision buffer), a negative reward is assigned to the system, because collision is assumed.
- $r_i^{\text{agent coll}}(\boldsymbol{s}),\ \forall i$, is the negative reward due to collision involving agent i. If two agents are closer together than a predefined number of cells (agent collision buffer), then a collision is detected and a negative reward is assigned to the system.
- $r^{\text{connect}}(\boldsymbol{s})$ is the negative reward due to loss of connectivity. The connectivity of the system is quantified using the second smallest eigenvalue λ_2 of the Laplacian matrix $L = D - A$ of the system, as discussed in [3]. The Laplacian matrix depends on the adjacency matrix $A = [a_{i,j}]$ of the system, where $a_{i,j}$ determines the connectivity between

agents i, j. $a_{i,j}$ is assumed non-binary, but spanning the range $(0, 1)$ depending on the distance between agents i, j (the distance is used to evaluate a Gaussian function centered at 0 to get the $a_{i,j}$). As distance increases the connectivity $a_{i,j}$ decreases. The value of $a_{i,j}$ results by evaluating a truncated Gaussian function at the distance value between the agents. The cut-off values of the truncated Gaussian represent the maximum distance for which connectivity is achieved (connectivity buffer), while the standard deviation of the Gaussian represents the decrease in connectivity quality with increasing distance.

F. Solution Method

We follow a centralized-learning-decentralized-execution framework. There are two distinct phases in our approach, following opposite methodologies of centralized versus decentralized.

1. Training Phase

To learn the policy, we use a centralized framework. This is motivated by the problem structure necessary to use the package POMDPs.jl. In order to use POMDPs.jl, a POMDP formulation is necessary rather than a Dec-POMDP. In our case, a POMDP formulation entails a centralized framework, with joint states and actions for the full multi-robot system. This joint formulation (with $|\mathcal{S}| = (n \times n)^{|\mathscr{D}|}$ and $|\boldsymbol{A}| = 9^{|\mathscr{D}|}$) is much larger than the independent agents' case (with $|\mathscr{D}|$ agents, each with $|\mathcal{S}| = (n \times n)$ and $|\boldsymbol{A}| = 9$). However, the joint formulation provides a collaborative multi-robot policy. During the training phase we calculate a near-optimal policy that maps observation tuples $(o_1, ..., o_{|\mathscr{D}|})$ to joint actions $(a_1, ..., a_{|\mathscr{D}|})$. To solve the POMDP we use the QMDP algorithm, which maintains an alpha vector for each joint action and determines the best action by finding the corresponding alpha vector that maximizes the inner product with the current belief. Given that the state uncertainty is concentrated around the true state, there is no use for information-gathering actions. Therefore, QMDP is expected to perform adequately [10].

2. Execution Phase

Given that our goal is to maintain connectivity or recover from a connectivity loss, a centralized deployment of our algorithm serves no use, since it assumes communication with a central processing unit, which is only possible if connectivity exists. Therefore, we aim for a decentralized execution. We begin by sharing the learned policy with all follower robots. Each individual follower robot i uses the learned policy by inputting its own maintained belief. This belief is updated using the independent observation tuple $(\hat{o}^i_1, ..., \hat{o}^i_{|\mathscr{D}|})$ available at robot i.

IV. Simulation Results

To test our proposed approach, we implement two scenarios. In scenario 1, we consider full observability of the true system state. In other words, we simplify the POMDP formulation to an MDP formulation. In scenario 2, we further introduce observation uncertainty, as discussed in Section III.D. In both scenarios, we assume a centralized-training-decentralized-execution approach, as discussed in Section III.F.2. Figure 4 shows the environment of the simulation for both scenarios as well as the pre-defined path of the leader for both scenarios. We assume one leader robot and one follower robot and the obstacles as shown.

Figure 5 shows two examples of a two-dimensional slice of the policy space. These examples correspond to the case that the leader chooses to remain at its current state and indicate the best action of the follower for every possible position on the map. Recall that the policy maps each belief $\boldsymbol{b} = [b(\boldsymbol{s_1}), ..., b(\boldsymbol{s}_{(\boldsymbol{n\times n})^{|\mathscr{D}|}})]$, to an action $\boldsymbol{a} \in \boldsymbol{A}$ with $|\boldsymbol{A}| \in 9^{|\mathscr{D}|}$. We have $|\mathscr{D}| = 2$ for the single leader and single follower case that we are considering. To make the slice, we fix the state and action of the leader. Call Γ_l the set of alpha vectors that are consistent with the leader performing a specific action a_l. We estimate the value function via Eq. (5).

$$U(s) = \max_{\alpha \in \Gamma_l} \alpha^\top b(s). \tag{5}$$

Since the rewards are of varying magnitude, we use a logarithmic scale to visualize the estimated value function Eq. (6).

$$\rho = \log(-U + \epsilon) \tag{6}$$

where ρ is the scaling (can be interpreted as the penalty) and ϵ is a small number (10^{-10} in our case) for numerical stability in plotting the cases with 0 rewards. This method produces a two-dimensional slice since we map each follower state $s_f = (x_f, y_f)$ to an action a_f and color code this state with the scalar value function.

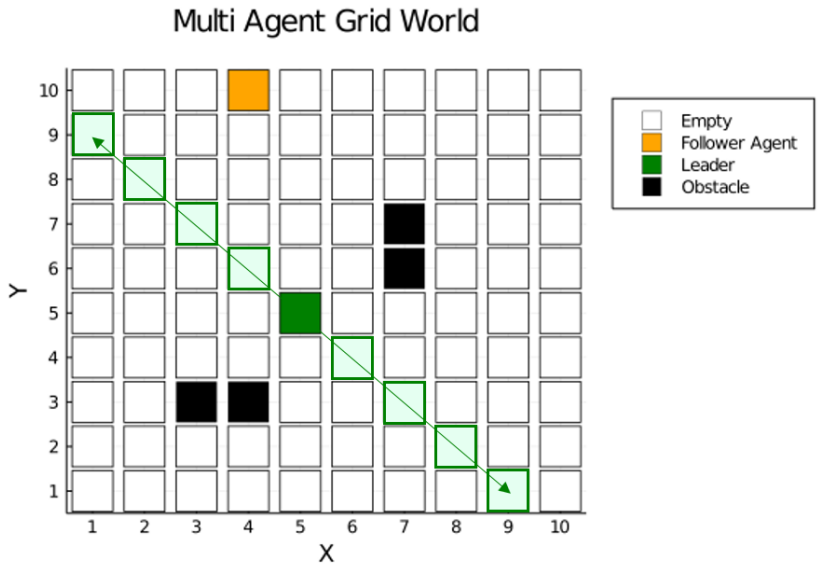
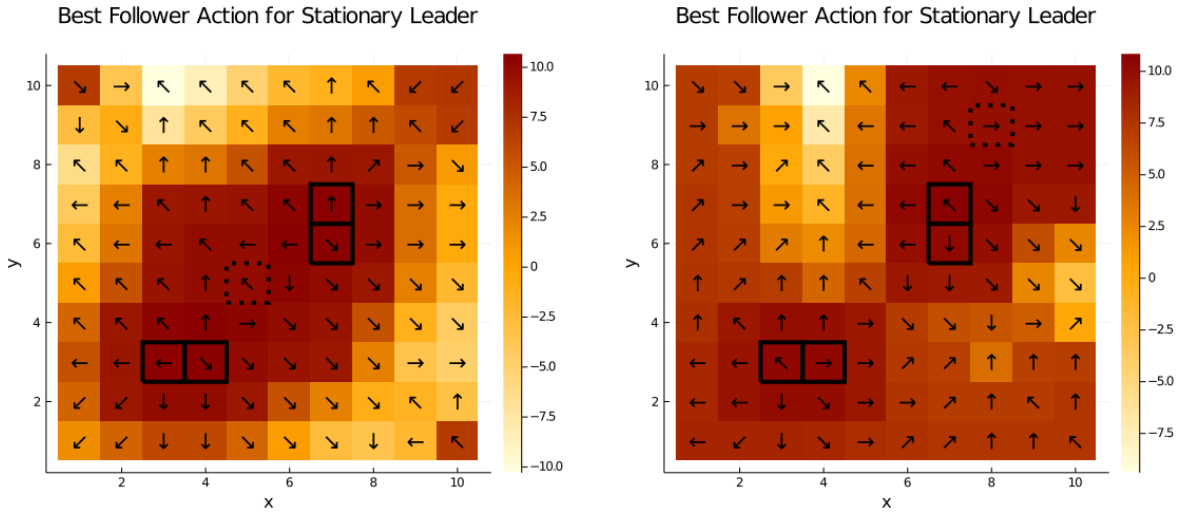


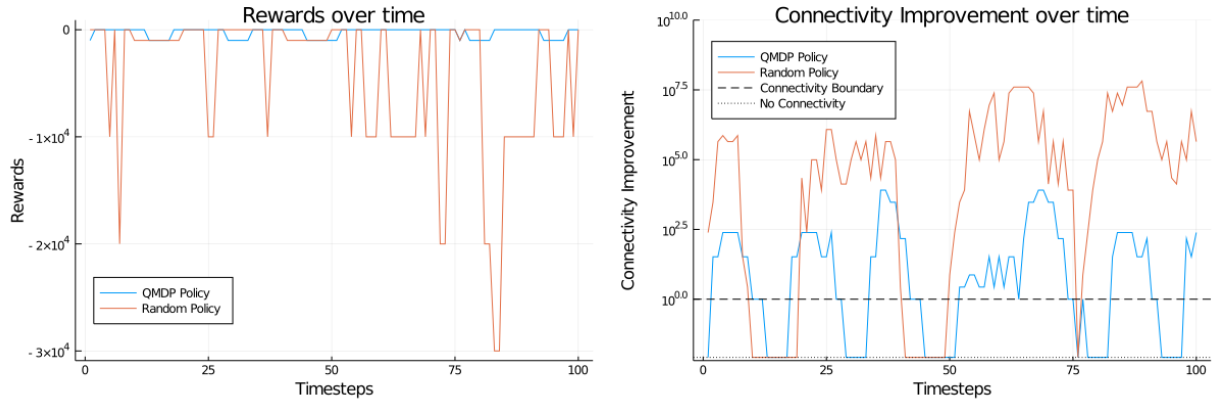
Fig. 4 World view of the multi agent system. The initial states of the follower and leader are indicated in orange and green, respectively, and the pre-defined path of the leader is outlined in green.

Figure 5 demonstrates how the follower robot tries to move away from the leader robot and the obstacles to avoid a potential collision. It thus reaches the border of connectivity (as described by the number of cells between agents) and tries to remain there. In this way, it is able to communicate and remain safe.



(a) Leader at (5, 5) **(b) Leader at (8, 9)**

Fig. 5 Policy for the follower agent conditioned on the leader's state and action. The arrows indicate the action that the follower would take if it believes it is at that state. The color scale is the logarithm of the negative estimated value function. Darker shades of red indicate large negative utility. The follower would prefer to move away from the dark red regions towards the light yellow regions. The cell with dotted borders represents the location of the leader. The cells with solid borders represent the obstacles or hazards in the environment, matching Figure 4.



(a) Time-Series of Rewards in the MDP Scenario. **(b) Time-Series of Connectivity in MDP Scenario.**

Fig. 6 Simulation Results for the MDP Scenario.

A. Scenario 1: MDP with Hazardous Environment

The first scenario is a simplified example with perfect observations – reducing the problem to an MDP. The map is as shown in Figure 4 and the policy learned by the QMDP solver follows the general idea in Figure 5. With perfect observations, the environment was artificially made more hazardous to examine the effect on policy learning. In this scenario, the object collision and agent collision buffers are Euclidean distances of 1 and 2, respectively (e.g. a collision would be detected between agents at $(4, 4)$ and $(5, 6)$). This environment can be classified as harsh, given that the obstacles, along with their buffer, cover a significant portion of the total space, while agents collide in many different settings. In this scenario, we assume a connectivity buffer of 6. This results in connectivity occurring in the largest portion of the environment. Therefore, we expect that the main challenge is not connectivity maintenance, but rather collision avoidance.

The time-series of rewards for our learned policy using QMDP and for a random policy are shown in Figure 6. For the MDP scenario, the rewards associated with obstacle collision, agent-to-agent collision, and loss of connectivity are -10000, -10000, and -1000, respectively. The rewards collected for each policy indicate that the QMDP policy induces far fewer collisions than a random policy, while still encountering some difficulties due to the large collision buffers imposed on the system.

The system connectivity for both policies is shown in Figure 6. In this plot, we measure the connectivity improvement by comparing the current algebraic connectivity (λ_2) to the algebraic connectivity value achieved when the robots are at the maximum allowed distance for connectivity ($\lambda_{2,\text{boundary}}$). The connectivity boundary line is at value 1. This represents the lowest threshold for connectivity. If the agents are further away than the allowable distance, then there is no connectivity. We denote the connectivity improvement $\lambda_{2,\Delta}$ as calculated in Eq. (7). Cases of $\lambda_{2,\Delta} \gg 1$ indicate stronger connectivity and are desirable. Cases of $\lambda_{2,\Delta} < 1$ indicate that $\max(\lambda_2, \epsilon) = \epsilon$, so there is no connectivity.

$$\lambda_{2,\Delta} = \frac{\max(\lambda_2, \epsilon)}{\lambda_{2,\text{boundary}}} \tag{7}$$

The connectivity of the random policy is superior to that of the QMDP policy; however, this is partly due to the frequent agent-to-agent collisions occurring with the random policy. It would not be an apt comparison to say that the random policy performs better, without acknowledging that it severely fails in the collision avoidance metric. Regardless, the QMDP policy manages to maintain connectivity to some extent, even in this difficult and hazardous scenario. The qualitative metrics for the QMDP scenario are shown in Table 1, where it can be seen that the QMDP policy outperforms the random policy in terms of cumulative rewards but not the percentage of time with connectivity.

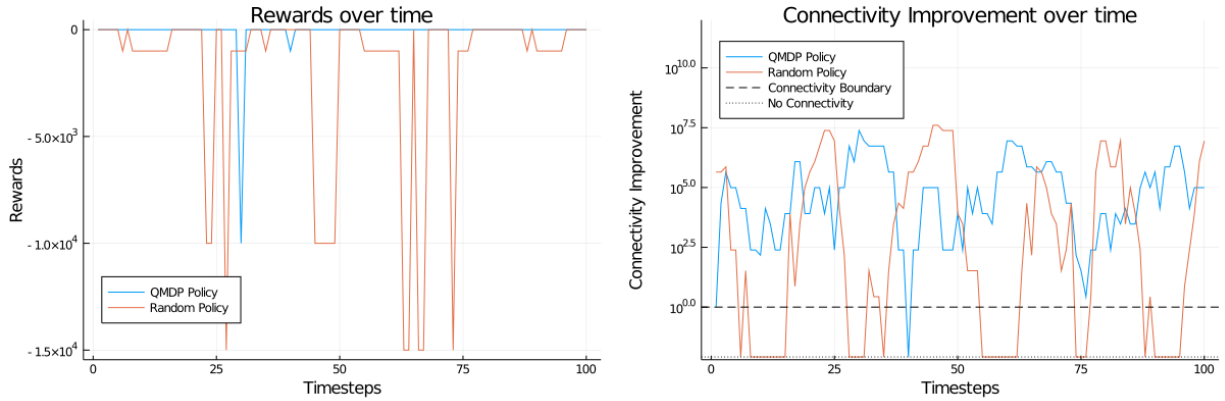
Table 1 Quantitative Metrics for MDP Scenario

Metric	QMDP	Random
Cumulative Sum of Rewards	-29000	-440000
Percentage of Time Connected	71%	80%

B. Scenario 2: POMDP with Relaxed Hazards

This scenario uses the same obstacle map as the previous scenario, but with relaxed constraints on the obstacle and inter-agent collision buffers to 0 and 1, respectively. Additionally, measurement noise has been introduced to corrupt agent observations, such that the perfect observation is obtained only 46.6% of the time. The same leader path is followed as in the previous scenario (back-and-forth from corner to corner) and the agent selects actions to maintain connectivity while avoiding collisions. Figure 7 shows the time-series of rewards collected by the agent using the QMDP trained policy and a random policy. For the POMDP scenario, the rewards associated with obstacle collision, agent-to-agent collision, and loss of connectivity were changed to -15000, -10000, and -1000, respectively.

It can be seen that the agent entirely avoids collisions in the POMDP scenario, whereas the random policy is involved in multiple agent-object and agent-to-agent collisions. The connectivity plot for the agents using both random and QMDP policies is shown in Figure 7.



(a) Time-Series of Connectivity in POMDP Scenario. **(b) Time-Series of Connectivity in POMDP Scenario.**

Fig. 7 Simulation Results for the POMDP Scenario.

The QMDP policy significantly outperforms the random policy by maintaining connectivity at nearly all times. Showing improvement over a random policy in connectivity maintenance and collision-avoidance are not necessarily significant findings – comparison against expert knowledge or strictly connectivity-maintenance or collision-avoidance policies may provide better metrics. Regardless, the learned policy is able to make strides towards achieving the goals set forth for the agent, showing promise for learning-based methods in this application. Overall metrics for the QMDP-learned policy in terms of connectivity and collision-avoidance are shown in Table 2. It is shown that the QMDP policy significantly outperforms the random policy as well as the policy from Section IV.A

Table 2 Quantitative Metrics for POMDP Scenario

Metric	QMDP	Random
Cumulative Sum of Rewards	-11000	-19200
Percentage of Time Connected	99%	32%

These improvements in both cumulative rewards and connectivity maintenance can be attributed to the significantly less treacherous environment that the POMDP policy was trained on. It should be noted that the additional free space in this environment allows the agent to dedicate more priority to connectivity rather than collision avoidance due to the substantially lower collision risk.

V. Conclusion

In this work, we propose a learning-based solution for the problem of connectivity maintenance in a multi-robot system. We adopt a POMDP formulation and assume a discretized environment and state-action space. We perform a centralized-learning phase using the POMDPs.jl package and deploy the learned policy in a decentralized manner. We test our learned policy in two scenarios involving leader and follower robots and observe that the learned policy indicates taking reasonable actions for the problem at hand, while performing better than a random policy in both objectives for the POMDP scenario.

VI. Future Work

At first, the discretization of the state and action space leads to serious limitations in the performance of the algorithm, since performance improvement can only be reached through a finer discretization, which adds computational complexity. Therefore, switching to a method that can deal with continuous state and action spaces (Deep RL) might lead to better results. Also, the assumed discretization leads to an exponential growth of the action and state space as the number of robots increases, which indicates that our algorithm is not scalable. To tackle this issue, we may use the approach found in [6], where Deep RL is used to determine the team-level policy that is a concatenation of robot-level policies, which are dependent only on the local observations. This allows for easy decentralized execution and learning without the exponential growth in the state and action space, since the Deep RL module maps local observations to actions directly. As a different direction, we may receive formal guarantees for the connectivity maintenance, by including constraints on the formulation, rather than integrating connectivity in the reward function. We can also use a different framework for the problem, by applying a Dec-POMDP formulation rather than a POMDP one. Finally, we conclude that a random policy is not the most apt comparison against the learned policy, and that comparison with a dedicated obstacle avoidance or connectivity maintenance policy may yield more informative findings to characterize the performance of our policies.

VII. Contribution of Team Members

All members actively participated in developing the problem formulation, directing the solution approach, programming the code base, and producing this report. With respect to the software development, Bradley worked on writing the code for the probabilistic transition functions, observation functions, action representations, and simulation code. Daniel worked on the state representation, visualization, and simulation code. Alexandros worked on the code for the reward function and connectivity maintenance. Also, all of us were involved in writing the code for solving the resulting POMDP and brainstorming the simulation scenarios. Alexandros drafted the initial, full final report. Bradley, Daniel, and Alexandros then worked together to improve the draft to produce the final version. Overall, the necessary tasks were equally divided among the team.

References

[1] Yang, P., Freeman, R., Gordon, G., Lynch, K., Srinivasa, S., and Sukthankar, R., "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, Vol. 46, No. 2, 2010, pp. 390–396. https://doi.org/10.1016/j.automatica.2009.11.012.

[2] Sabattini, L., Chopra, N., and Secchi, C., "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, Vol. 32, No. 12, 2013, pp. 1411–1423. https://doi.org/10.1177/0278364913499085.

[3] Shetty, A., Hussain, T., and Gao, G., "Decentralized Connectivity Maintenance for Multi-Robot Systems Under Motion and Sensing Uncertainties," *Proceedings of the Institute of Navigation GNSS+ Conference*, St. Louis, Missouri, 2021, p. 14.

[4] Minghao, L., Yingrui, J., Yang, K., and Hui, C., "Decentralized Global Connectivity Maintenance for Multi-Robot Navigation: A Reinforcement Learning Approach," *CoRR*, Vol. abs/2109.08536, 2021. URL https://arxiv.org/abs/2109.08536.

[5] Lin, J., Yang, X., Zheng, P., and Cheng, H., "Connectivity Guaranteed Multi-robot Navigation via Deep Reinforcement Learning," *CoRL*, 2019.

[6] Lin, J., Yang, X., Zheng, P., and Cheng, H., "End-to-end Decentralized Multi-robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning," *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 2493–2500. https://doi.org/10.1109/ICMA.2019.8816208.

[7] Huang, W., Wang, Y., and Yi, X., "Deep Q-Learning to Preserve Connectivity in Multi-Robot Systems," Association for Computing Machinery, New York, NY, USA, 2017. https://doi.org/10.1145/3163080.3163113, URL https://doi.org/10.1145/3163080.3163113.

[8] Gupta, J. K., Egorov, M., and Kochenderfer, M., "Cooperative Multi-agent Control Using Deep Reinforcement Learning," *Autonomous Agents and Multiagent Systems*, edited by G. Sukthankar and J. A. Rodriguez-Aguilar, Springer International Publishing, Cham, 2017, pp. 66–83.

[9] Oliehoek, F. A., "Decentralized POMDPs," *Reinforcement Learning*, Vol. 12, edited by M. Wiering and M. van Otterlo, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 471–503. https://doi.org/10.1007/978-3-642-27645-3_15, URL http://link.springer.com/10.1007/978-3-642-27645-3_15.

[10] Kochenderfer, M. J., Wheeler, T. A., and Wray, K. H., *Algorithms for decision making*, Massachusetts Institute of Technology, 2022.

[11] Fernandez, F., and Parker, L. E., "Learning in large cooperative multi-robot domains," *International Journal of Robotics and Automation*, 2001.

[12] Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K., and Kochenderfer, M. J., "POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty," *Journal of Machine Learning Research*, Vol. 18, No. 26, 2017, pp. 1–5. URL http://jmlr.org/papers/v18/16-300.html.