# Improving Scalability in Multi-Robot Systems with Abstraction and Specialization

Sha Yi

CMU-RI-TR-19-46

May 2019

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Katia Sycara, Chair
Stephen Smith
George Kantor
Wenhao Luo

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

**Keywords:** Multi-agent, planning, control, POMDP, connectivity

*For my parents*

# Abstract

Planning and controlling multi-robot systems is challenging due to high dimensionality. When the number of robots increases in the system, the complexity of computation grows exponentially. In this thesis, we examine the scalability problem in planning and control of the multi-robot system. We propose two ideas to improve the scalability of multi-robot systems: abstraction and specialization.

For planning with abstraction, we propose a Hybrid Hierarchical Partially Observable Markov Decision Processes (POMDPs) structure for improved scalability and efficiency in an indoor environment that creates abstract states of convex hulls over the grid environment. We focus our application on the problem of pursuit-evasion with multiple pursuers and one evader whose location is unknown if not visible to the pursuers. This approach is scalable that it significantly reduces the number of states expanded in the policy tree to solve the problem by abstracting environment structures.

Specialization is important for systems with a large number of robots. Connectivity maintenance is essential for collaborative behaviors since robots need to communicate with neighbor robots to share states and information. It is inefficient for a robot to both attend to task behaviors and also maintain connectivity. Therefore, we introduce the idea of *connection robots*, whose goal is to maintain connectivity of the robot team by correcting the topology of the connectivity graph so as to provide flexibility for the task robots to perform task behaviors. We propose a scalable distributed approach of topology correction that is able to guarantee a faster convergence rate with response to dynamic environment and tasks.

## Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Katia Sycara for the continuous support of my Master study and research for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and preparing for this thesis. Although being always busy, she is always available for discussion when I am stuck or have new ideas. I would not be able to accomplish my achievement without her help.

Besides my advisor, I would like to thank the rest of my thesis committee for their insightful comments and encouragement, but also for the questions that widened my research from various perspectives.

I thank my fellow labmates in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last two years. Special thank to Wenhao Luo, who has answered all my weird questions and provided valuable insights into my research problems.

I would also like to thank my boyfriend Gengshan Yang for getting me awesome food when I get depressed, and being a wonderful companion inside and outside academic studies.

Last but not least, I would like to thank my parents for supporting me spiritually throughout my pursuit of an academic career and my life in general.

# Contents


**1 Introduction** **1**

**2 Related Work** **5**
2.1 Multi-robot Coordination and Pursuit-evasion . . . . . 5
2.2 Planning under Uncertainty and Abstraction . . . . . 5
2.3 Connectivity Maintenance . . . . . 6

**3 Plan with Abstraction** **7**
3.1 Problem Definition . . . . . 7
3.2 Hybrid Hierarchical POMDP . . . . . 8
3.2.1 Multi-agent POMDP . . . . . 8
3.2.2 Modeling of Environments using Convex Hulls . . . . . 8
3.2.3 Hybrid Hierarchical POMDP Structure . . . . . 9
3.2.4 Transitions and Rewards . . . . . 11
3.3 Experiment Results . . . . . 14

**4 Control with Specialization** **17**
4.1 Problem Definition . . . . . 17
4.2 Topology Correction . . . . . 18
4.2.1 Graph Laplacian and Convergence . . . . . 18
4.2.2 Weighted Rendezvous . . . . . 19
4.2.3 Weighted Flocking . . . . . 22
4.2.4 Weighted Behavior Combination . . . . . 23
4.3 Results . . . . . 23

**5 Conclusion and Future Work** **29**

**Bibliography** **31**

# List of Figures


3.1 Modeling the grid world. (a) The world map is discretized into grids. (b) Overlapping convex hulls are defined based on the free space in the map. (c) The intersections of convex hulls form borders. . . . . 9
3.2 The Hybrid Hierarchical POMDP Structure. The structure consists of the base MDPs for the cases where the evader is visible to the pursuers, the abstract POMDPs for the evader states that are not directly observable, and the transition states bridging between the base MDPs and abstract POMDPs. . . . . 10
3.3 Environment maps with different indoor structures . . . . . 15
3.4 The number of nodes expanded in our HHPOMDP and a standard POMDP . . . 15
3.5 Results with Map 4 with different numbers of robots . . . . . 15
3.6 Environment maps with different sizes . . . . . 16
3.7 Result with 2 robots on maps of different size . . . . . 16

4.1 Two example scenarios for a system with robots allocated with tasks and connection robots. Left: connection robots need to keep up with task robots to avoid being left behind; Right: connection robots need to spread out so as to provide flexibility for the task robots to execute tasks. . . . . 18
4.2 The moment when vertex $v_j$ is moving to $v_k$ resulting in $v_i$ and $v_j$ getting disconnected from each other. . . . . 20
4.3 An example when the task robot is trying to move to an assigned location but the connection robots are cluttered along the way. . . . . 22
4.4 (a) size 30×36, (b) size 30×36, and (c) size 27×39, are the three maps for the experiments. The black area is the walls or obstacles in the environment, and the white area is the free space. . . . . 24
4.5 (a) Average computation time for each iteration; (b) Average eigenvalues (c) Variance of the robot locations at each time of convergence; (d) Average distance to goal for the task robots at each time of convergence . . . . . 24
4.6 Red robots: task robots, blue robots: connection robots, light blue circles: goal locations. The process of $N = 10$ robots executing task 2 and 3 in Map 4.4a with no connection controller. (a)-(b) Weighted behavior combination: the task robots are able to reach all the goal locations and the connection robots are able to keep up with the task robots till the end; (c)-(d) No connection controller: robot failing the task due to connectivity constraints; (e)-(f) Weighted rendezvous: connection robots blocking the task robots; (g)-(h) Weighted flocking: some robots were left behind; (i)-(j) Rendezvous: connection robots blocking the task robots. . . . . 25

4.7 Red robots: task robots, blue robots: connection robots, light blue circles: goal locations. The process of $N = 10$ robots executing task 1 and 3 in Map 4.4b with no connection controller. (a)-(b) Weighted behavior combination: the task robots are able to reach all the goal locations and the connection robots are able to keep up with the task robots till the end; (c)-(d) No connection controller: robot failing the task due to connectivity constraints; (e)-(f) Weighted rendezvous: perform the same as in weighted behavior combination; (g)-(h) Weighted flocking: some robots were left behind; (i)-(j) Rendezvous: robot failing the task due to connectivity constraints. . . . . 26

# Chapter 1

# Introduction

Multi-robot systems have been widely studied for their ability to perform collective behaviors to accomplish complicated tasks, for example, environment exploration [4], cooperative sampling [17], pursuit and evasion [13], search and rescue [16]. Planning and controlling multi-robot systems have been challenging due to the high dimentionality of state space, which would increase computation exponentially with the increasing number of robots within the system. In this thesis, I will present two ways of handling the high dimentionality problem: abstraction and specialization.

For the abstraction, we propose a hybrid hierarchical structure for planning under uncertainty. The application scenario we consider is the pursuit-evasion problem in indoor environments, which have received significant attention owing to the complex structure that makes the problem more difficult. We also consider multiple pursuers as utilizing collaborative peers for pursuit increases the probability of catching the evader. However, (i) the limited sensor view of pursuers caused by stationary obstacles and (ii) the coordination with peer pursuers make the problem complex and difficult to model using simple representations.

One of the early work [13] formulated the problem as a multi-agent graph search problem. Although its efficient performance and scalability for locating a target object, the graph search approach is not sufficiently rich to model the uncertainty from the target that could disappear from the sensing range of the pursuers after they observe the target. Partially Observable Markov Decision Processes (POMDPs) can incorporate the uncertainty caused by the target that could disappear from the pursuers' view since POMDPs allow the state variables that cannot be directly observed. For example, the pose of the target may not be directly observed as it can move freely while it is not in the line of sight of the pursuers. Although POMDPs is capable of modeling such targets, the state space of POMDPs grows exponentially with the number of robots which makes the approach intractable.

Thus, we introduce a scalable decision-making framework for the multi-robot pursuit-evasion problem, which is *Hybrid Hierarchical Partially Observable Markov Decision Processes (HH-POMDPs)*. Our framework creates an abstraction of the environment and reduces the state space for improved scalability. We define three planning states in our framework: *Base MDP states*, *Transition states*, and *Abstract POMDP states*. The base MDPs are defined for the cases where the evader is visible to the pursuers. The abstract POMDPs provide decision-making processes for the pursuers if the evader states are not directly observable as the evader can be out of the

sensing areas of the pursuers. The transition states provide an interface to switch between the base MDPs and abstract POMDPs. For the abstract POMDP state, we model the environment with convex hulls to reduce the size of state space. The base MDP states are the states from the grid world directly. The algorithm terminates when one of the pursuers is at the same location in the grid world as the evader.

Compared with standard POMDP approaches, our HHPOMDP method creates a hierarchical structure that utilizes environmental characteristics to reduce the complexity of the state space. Our approach of modeling the map with convex hulls and borders reduces the size of state space significantly compared with simple grid world models, while still maintain sufficient information for planning as well as incorporating the uncertainty of the target.

In the rest of this paper, we will define the problem formally in Sec. 3.1, present the detailed algorithm in Sec. 3.2, and finally present experimental results and comparisons in Sec. 3.3.

With a growing number of robots available for performing tasks, specialization is helpful for controlling the robots to achieve their goals. In continuous space and real-world applications, each robot within the system usually has a limited range of communication [21] and is only able to exchange information with its neighbors in the connectivity graph. Maintaining connectivity of the whole system is essential [14] since the collaborative behaviors rely highly on the connection between robots [29] and it takes extensive amount of work to restore the connection once it is lost [19]. In this network robotics system, distributed algorithms, where each robot reasons and controls using only the local neighbor information, is also important for the performance and scalability [29].

Most work focuses on maintaining connectivity based on existing control laws [11, 29], or study the connectivity from a given behavior or state [24, 28]. These are based on the assumption that every robot within the system is assigned a set of behaviors to perform in sequence [22] or in parallel. Due to connectivity constraints or environmental limitations, some of the robots might not be able to perform the desired behavior. For example, when some robots are assigned the behavior of flocking north and the other robots are assigned to flock south, the system could end up with some robots staying in the middle to maintain connectivity of the whole system. However, it may be more efficient to add robots whose primary task is to maintain the connectivity of the whole system. In other words, these robots would not have any task behavior assigned to them. We define this kind of robot as *connection robots*. Accordingly, we define the robots with assigned behaviors to be *task robots*. With support from *connection robots*, the task robots with assigned behavior may have more flexibility in achieving their goals without being overly constrained by connectivity maintenance.

In this part, our main goal is to design and analyze the controller for connection robots to maintain a flexible connectivity graph with *provably faster convergence rate* of the whole system, so as to support robots with assigned behaviors to perform as desired. The challenges for connection robots are 1) keeping up with the task robots; 2) avoiding blocking the task robots, and 3) providing fast convergence rate. We will present *weighted rendezvous*, *weighted flocking*, and *weighted behavior combination* that combines weighted rendezvous and weighted flocking to deal with the above-mentioned challenges. We will show, both theoretically and experimentally, that our method is able to provide flexible connectivity for the task robots to perform their assigned behavior by distributedly correcting the topology of the connectivity graph.

In this thesis, I will present the results of both methods on various scenarios and maps to

show that both algorithms are efficient and scalable.

# Chapter 2

# Related Work

## 2.1 Multi-robot Coordination and Pursuit-evasion

Pursuit-evasion has received much attention for decades. Guibas [9] formulated the problem in a way that the final goal is to "see" the evader and also provided a guarantee for capture in polygonal environments. For the goal in similar environments, some later work represented the environment as a graph. Isler et al. [15] proposed building a roadmap for the environment so as to discretize the continuous space. However, this still results in a large state space because the roadmap does not compactly describe the environment thus produces a relatively large number of edges and vertices compared to the size of the environment.

Hollinger et al. [12, 13] proposed coupled and decoupled coordination strategies for multiple pursuers. The coupled coordination strategy includes a centralized planner that searches all possible paths on the graph. This strategy takes the future positions of other pursuers into account. The first robot plans on the original graph and the rest plan on the time-augmented graph which adds time as an additional variable. For the decoupled coordination, each robot plans for itself, assuming others' paths are fixed. They model the environment with convex hulls, then build a graph with vertices as the convex cells based on the discretization. However, their method models all the free space in the map by a set of non-overlapping convex hull so cannot capture the case where the pursuer is able to see multiple convex hulls from an overlapping region of those convex hulls. For example, at a corner connecting two corridors in different directions, the pursuer can see both corridors without moving to each of the convex hulls representing each corridor. The position of the target/evader is maintained by a belief vector and its transition is based on a dispersion matrix, which is constructed according to the area of adjacent cells. The policy is computed by searching on the graph. However, since the framework aims to find the location of the target but does not catch the target physically, the target might evade and disappear from the sight of pursuers, which is not capable of modeling the target in our problem.

## 2.2 Planning under Uncertainty and Abstraction

Many probabilistic search problems, which include moving targets or dynamic environment, can be formulated as Markov Decision Processes (MDPs). If the moving target is not always

visible to the pursuer, it can be formulated as a Partially Observable Markov Decision Process (POMDP). An early work uses reinforcement learning with POMDP [1] for multi-agent planning on a grid world. In that work, each agent learns its own policy with a credit assignment method. Hollinger [13] proposed to solve the multi-robot search problem with joint space of multiple pursuers in POMDP, but the state space grows exponentially as the number of pursuers increases. A recent work on abstract MDPs [8] introduced abstraction to the state space for top-down policy search for MDP. However, the joint space of POMDPs is still intractably large. Ong [25] provided the idea of mixed observability, a special class of POMDPs. This approach separates the fully and partially observable components of the state and leads to a faster planning algorithm. Our work is motivated by both abstraction and mixed observation. We introduce a planning structure on top of these special structures of POMDPs.

## 2.3 Connectivity Maintenance

Multi-robot systems can accomplish collaborative tasks with predefined control laws towards the goal region such as flocking strategy [23, 30] or from a sequence of behavior library [22]. Such collaborative performance relies on communication between neighbors in the connectivity graph [29] to exchange robot states and information.

Connectivity maintenance in multi-robot systems with predefined control laws has been extensively studied in the literature [20, 28, 29]. Barrier certificate [3] initially proposed to serve as avoiding collisions between robots, can also be used to formulate constraints on robots to keep connected within a limited range. To measure the connectivity of the communication graph, [7] stated the relationship between the second smallest eigenvalue of the Laplacian matrix of the graph. This is further discussed in detail in [24] which also stated the relationship between the convergence rate and stability of the robotics system. In [23, 24], it is shown that the convergence rate of rendezvous and flocking is directly related to the second smallest eigenvalue of the Laplacian matrix, which is also determined directly by the degrees of the graph vertices, as well as other topological properties.

# Chapter 3

# Plan with Abstraction

## 3.1 Problem Definition

We study the pursuit-evasion problem in indoor environments where $k$ pursuers search one evader. The goal of the pursuers is to hold the evader physically in close proximity. The configuration of the environment is known to the pursuers but not necessarily known to the evader. We assume a static environment and discretize the environment into a grid world. For each time stamp, both a robot (pursuer) and a target (evader) can move to an adjacent cell on the grid map, or choose to stay in the same cell, which means they have the same base action space in the grid world. We assume the line-of-sight model of the robotic pursuers, which means the robots are able to see the target if there is no obstacle or wall blocking the sight, otherwise they do not directly know the target location.

Let the state of the $i$th robot $R_i$, $i = 1, 2, \cdots, k$, at time $t$ be $s_{R_i}(t)$ and the state of the target $T$ at time $t$ be $s_T(t)$. We assume a 4-connected grid so the action space of both robot and target is $A = \{N, E, W, S, Z\}$, which corresponds to move to North, East, West, South, and remain in the current cell (i.e., no-op). The transition by applying an action to a given state of a pursuer is deterministic, which indicates the probability of arriving the next state $s'_{R_i}$ from the current state $s_{R_i}$ is $p(s'_{R_i}, a, s_{R_i}) = 1$. Consider $A_v \subseteq A$ to be a valid set of actions for a given state which satisfies each action $a \in A_v$ and does not lead to collision to environment obstacles or walls. In our current scenario, we assume the target moves randomly with a uniform distribution over its action space. Therefore, we have the probability of action $a$ at a given state $s$:

$$p(a) = \begin{cases} \frac{1}{|A_v|}, & a \in A_v \\ 0, & a \notin A_v. \end{cases} \tag{3.1}$$

The transition state is defined as one where a pursuer catches the evader, which means that the pursuer and the evader are in the same grid cell at the same time, i.e., $s_{R_i}(t) = s_T(t)$.

The objective of the pursuers is to find an optimal policy $\pi^\star$ that minimizes the expected time of capturing the evader. Since the reward is discounted, the goal is equivalent to maximizing the expected reward for the pursuers, which is:

$$\pi^\star = \arg\max_{\pi} V(s). \tag{3.2}$$

## 3.2 Hybrid Hierarchical POMDP

In this section, we review a standard Multi-agent POMDP with a direct joint state space. Then we introduce our method of modeling the environment with convex hulls and borders. We also describe our HHPOMDP structure in detail and define the transition and reward functions accordingly.

### 3.2.1 Multi-agent POMDP

A *Partially Observable Markov Decision Process* (POMDP) describes a stochastic process where some of the state variables may not be directly observable [10]. The process is described by $(S, A, \Theta, T, O, R, \gamma)$, where $S$ is the state space, $A$ is the action set, $\Theta$ is the observation set, $T : S \times A \times S$ is the transition matrix given states, actions, and the next states, $O : S \times A \times \Theta$ is the probability of observations given states, actions and the observation tuple, $R : S \times A \times S$ is the reward given states and action transitions, and $\gamma$ is the discount factor. The goal is to compute an optimal policy $\pi^\star : S \times O \to A$ that maps from each state and observation pair to an action that maximize the expected discounted reward.

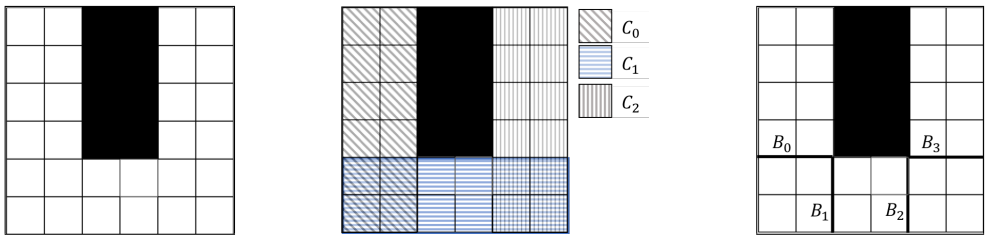
For multi-agent POMDP with $k$ agents [27], each state $S = S_1 \times S_2 \times \cdots \times S_k$, action set $A = A_1 \times A_2 \times \cdots \times A_k$, observation set $O = O_1 \times O_2 \times \cdots \times O_k$ are all joint state space of each agent. Similar with single agent POMDP, the multi-agent POMDP tries to compute an optimal joint policy $\pi^\star : (\pi_1^\star, \pi_2^\star, \cdots, \pi_k^\star)$ that each $\pi_i^\star : S_i \times O_i \to A_i$ maps from each state and observation pairs to a corresponding action that maximize the expected discounted reward.

### 3.2.2 Modeling of Environments using Convex Hulls

We convert the grid map of an indoor environment into a set of convex hulls. This representation of the environment reduces the size of the state space to a great extent but does not lose necessary information regarding the environment. By definition of convex hull, it is guaranteed that the target is within the sight of a pursuer.

With this abstraction of the environment, grid cells could be separated and grouped into regions, thus we could utilize the environment characteristics for planning and reduce the state space.

As shown in Fig. 3.1, the grip map of the environment is modeled in the following way: Fig. 3.1a shows the original environment map in grids where the black grid cells represent walls and obstacles and the white area represents the free space. We model the environment by grouping grid cells into convex hulls such that the number of convex hulls is minimized. A convex hull is denoted as $C_i$, which also will be used as a state variable in the state space of the evader. As shown in Fig. 3.1b, the convex hulls may have overlapping areas. The overlapping areas of the adjacent convex hulls form borders shown in Fig. 3.1c. A border is denoted as $B_i$, which will be used as a state variable of the state space of the pursuers. At each convex hull border $B_i$, a pursuer is able to directly observe the target in any of the adjacent convex hulls as proved in Proposition 1. In this present work, the convex hulls are generated through unautomated processes. However, it can be generated directly from the input map if we iterate through all the cells and use any standard convex hull finding algorithm, for example Jarvis's Algorithm [6].




(a) Grid Map (b) Convex Hull (c) Convex Hull Borders

Figure 3.1: Modeling the grid world. (a) The world map is discretized into grids. (b) Overlapping convex hulls are defined based on the free space in the map. (c) The intersections of convex hulls form borders.

### 3.2.3 Hybrid Hierarchical POMDP Structure

For multi-agent planning, the direct way for the optimal result is to plan on the joint state space of all robots as described in Sec. 3.2.1. However, the complexity grows exponentially with the number of robots. For multi-agent POMDP, the complexity is much higher than direct joint MDP since it is necessary to track history and record the belief space vector. Note that the complexity of POMDPs even with a single agent is PSPACE-complete [26].

We notice that when the target is not observable to any of the pursuers, its exact position on the grid map does not make a significant difference to the exploration plan of the pursuers because the pursuer needs to "see" the target (i.e., both parties are in the same convex hull) before catching. Instead, only the convex hull in which the target lies within influences the next location of pursuers. Thus, the state space which is partially observable to the pursuers could consist of the convex hull borders while the exact location of the target on the grid cells is omitted. On the other hand, when the target and pursuers are in the same convex hull, the location of the target is fully observable to the pursuers. Thus, it is not necessary to keep track of any observation history, which means that an MDP can be used to improve computational efficiency. Therefore, we propose a *Hybrid Hierarchical structure* of MDPs and POMDPs as shown in Fig. 3.2.

Specifically, consider $k$ pursuers (robots) and one evader in a grid world environment discretized into $N$ convex hulls and $M$ convex hull borders. We denote the location of the evader (target) with a belief space vector $b = [b_0, b_1, \cdots, b_{N-1}]$. As shown in Fig. 3.2, the nodes on top shows the abstract POMDP nodes, which is the joint state space of all pursuers. The nodes in the middle is transition states where all the belief vector of target location results in a specific convex hull. The nodes in the bottom are the base MDP nodes contains only the location of one pursuer and the target in the grid world. Let the state of robot $j$ be $S_{R_j}$ and the state of target to be $S_T$, the states in the abstract POMDP level is denoted as $S_i^p$, transition states as $S_i^t$, and the base MDP states are $S_i^b$.

The hybrid POMDP tree has a height of three. Level 1 is the joint abstract POMDP of all the pursuers' and target's state spaces $S_i^p : S_{R_1}^p \times \cdots \times S_{R_k}^p \times S_T^p$, where the target state space is the Cartesian product of belief of each individual states $S_T^p = \prod_{i=0}^{k-1} B_i$. Each robot has a starting state, the POMDP policy tree grows by applying all valid actions and observations $(a, o)$, and the belief space vector is updated accordingly. The belief space vector is maintained with a mixed

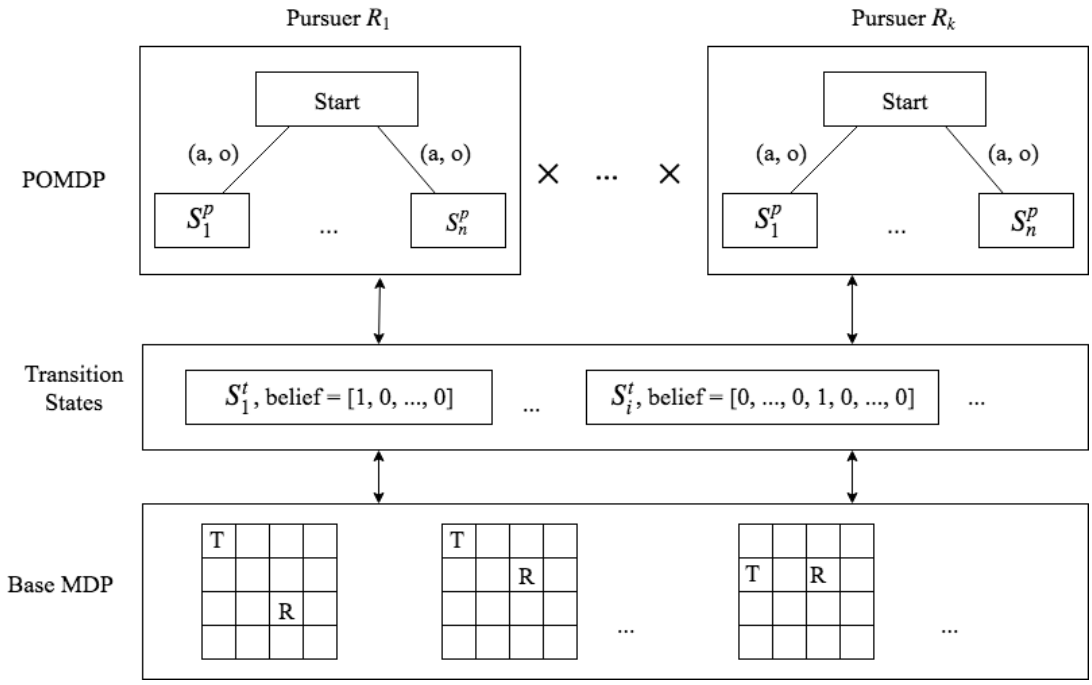



Figure 3.2: The Hybrid Hierarchical POMDP Structure. The structure consists of the base MDPs for the cases where the evader is visible to the pursuers, the abstract POMDPs for the evader states that are not directly observable, and the transition states bridging between the base MDPs and abstract POMDPs.

observability since we assume all robots are connected, the locations of all pursuers are known by all robots, and only the target location is partially observable. Level 2 is the transition state when the belief space shrinks to one convex hull, and the target is visible to one of the robots. Level 3 is the base MDP of the joint space of one robot $R_j$ and the target $T$, $S_i^b = S_{R_j}^b \times S_T^b$. When the target goes out of the line of sight of $R_j$, the state of the robot returns to Level 2 and if the belief space expands, it may return to Level 1.

### 3.2.4 Transitions and Rewards

#### Base MDP

The base MDP is fully observable and follows the standard Bellman equation for dynamic programming [2]:

$$V^*(s^b) = \max_{a^b \in A^b} \sum_{s^{b'}} T^b(s^b, a^b, s^{b'})[R(s^b, a^b, s^{b'}) + \gamma V^*(s^{b'})] \tag{3.3}$$

where $s^b$ is the current base state in the grid world, which is the coordinate on the grid map, $\gamma$ is the discount factor, $s^{b'}$ is the neighboring states of state $s^b$, and $T^b$ is the transition matrix and $A^b$ is the action space for the base MDP, which is the same as the action space in Sec. 3.1 that is $A^b = \{N, E, W, S, Z\}$.

Each base MDP state is the joint space of the target and the robot in the same convex hull $s^b = s_R^b \times s_T^b$. The action spaces of the robot and target are the same, which form the joint action space for the base MDP $A^b = a_R^b \times a_T^b$. Since the target moves randomly, the robot has a deterministic outcome for applying a given action, and the transitions of the robot and target are independent the transition probability is uniformly distributed over the all valid target neighboring states $S_T^b$:

$$T^b(s^b, a^b, s^{b'}) = p(s^{b'}|s^b, a^b) = \frac{1}{|S_T^b|}. \tag{3.4}$$

For the next state $s^{b'}$, it is possible that the target might disappear from sight. If this happens, the next state will go back to the transition state and might also further go back to the abstract POMDP level if the belief space vector changes. This procedure is described in Alg. 1.

Reward is only given to the final terminal state when one of the robot catches the target, which is when the target and a robot is in the same cell on the grid map:

$$R(s^b, a^b, s^{b'}) = \begin{cases} r, \ \exists j \in [0, k) \ s.t. \ s_{R_j}^b = s_T^b \\ 0, \ \text{otherwise.} \end{cases} \tag{3.5}$$

**Algorithm 1** Expand Base Node

**Input:** $c$: current node to be expanded, $A$: action set
**Output:** node set to be expanded

```
function EXPANDBASE(c, A)
    for a in A do
        n ← getNeighborNode(c, a)
        if not isVisible(c.robot, c.target) then
            n.type ← transition
        else
            n.type ← Base
        c.childList.append(n)
    return expandSet ∪ c.childList
```

**Algorithm 2** Expand Transition Node

**Input:** $c$: current node to be expanded
**Output:** node set to be expanded

```
function EXPANDTRANSITION(c)
    if not isVisible(c.robot, c.target) then
        n ← Node(c)
        n.type ← POMDP
        c.childList.append(n)
    else
        baseList ← getBaseGrid(c.belief)
        for b in baseList do
            n ← Node(b)
            n.type ← Base
            c.childList.append(n)
    return expandSet ∪ c.childList
```

**Algorithm 3** Expand POMDP Node

**Input:** $c$: current node to be expanded, $\Theta$: observation set, $A$: action set
**Output:** node set to be expanded

```
function EXPANDPOMDP(c, Θ, A)
    for (a, o) in (Θ, A) do
        n ← getNeighborNode(c, a, o)
        n.updateBelief()
        if max(n.belief) is 1 then
            n.type ← transition
        else
            n.type ← POMDP
        c.childList.append(n)
    return expandSet ∪ c.childList
```

### Transition State

A transition state represents a transition between the abstract POMDP and the base MDP. The transition states could be entered from either the base MDP or the abstract POMDP. When the belief space vector shrinks to one cell, i.e, $\exists b_i = 1$, and the target is visible to one of the robot, then the expansion process enters the transition states from the abstract POMDP. The expansion can also come from the base MDP when the target moves away from the line of sight of any robot, and in this case, the system needs to return to the abstract POMDP via those transition states. This procedure is described in detail in Alg. 2.'

### Abstract POMDP

The abstract POMDP has a mixed observability of the state space. The state space is the joint state space of all robot locations (convex hull borders) and the target location (convex hulls). For robot $R_j$, the state in the abstract POMDP level $s^p_{R_j}$ includes the convex hull borders which are fully observable. This belief space vector is updated during the expansion of the policy tree (Alg. 3) and will enter the transition state if one of the convex hull belief is one. For target $T$, the state space $s^p_T$ is the belief across convex hulls $b = [b_0, b_1, \cdots, b_{N-1}]$ which are partially observable. This formulation gives the mixed observability of state variables and is used to simplify computation in the transition and reward functions.

The state value update follows the traditional Bellman updates, but with the belief space vector [10]

$$V^*(b) = \max_{a \in A} \left\{ \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{o \in \Theta} \sum_{s \in S} T(o|s,a)b(s)V^*(\tau(b,o,a)) \right\} \tag{3.6}$$

where $R(s,a)$ is the reward function for action $a$ at a given state $s$, $T(o|s,a)$ is the transition probability of having observation $o$ given state $s$ and action $a$. The function $\tau$ represents the information state of the joint space of belief vector $b$, observation $o$ and action $a$. For our specific problem, the reward of the system only comes from the base MDP in the absorbing state where

the target and a robot are in the same cell. Thus, there is no reward in the updates from the abstract POMDP level itself.

Along with the mixed observability described above, we could simplify the value update based on the full observable states of the robots and only keep track of the belief vector over the target state space. The action space for the abstract POMDP is the joint action of all robots and target, which is the Cartesian product $A = A_T^p \times \Pi_j A_{R_j}^p$. Similarly, the robot state space is also the Cartesian product $s_R^p = \Pi_j s_{R_j}^p$. Since the robot transitions are deterministic and independent from each other as well as the target, the transition probability is only dependent on the target transitions, which is uniformly distributed over its adjacent convex hulls that the robot can move, and robots' visibility. Thus, the transition probability is

$$T(o|s,a) = T(o|s_T^{p\prime}, s_R^{p\prime}) \cdot T(s_T^{p\prime}|s_T^p) \cdot T(s_R^{p\prime}|s_R^p). \tag{3.7}$$

Then we have the following modified Bellman equation:

$$V^*(s^p) = \max_{a\in A} \left\{ \gamma \sum_{o\in\Theta} \sum_{s_T^{p\prime}} \sum_{s_R^{p\prime}} T(o|s_T^{p\prime}, s_R^{p\prime}) T(s_T^{p\prime}|s_T^p) T(s_R^{p\prime}|s_R^p) b(s_T^p) V(\tau(b,o,a)) \right\}. \tag{3.8}$$

The HHPOMDP system is solved by value iteration based on forward exploration from the root state [10], which will only search in all reachable belief states instead of the full span of belief vectors.

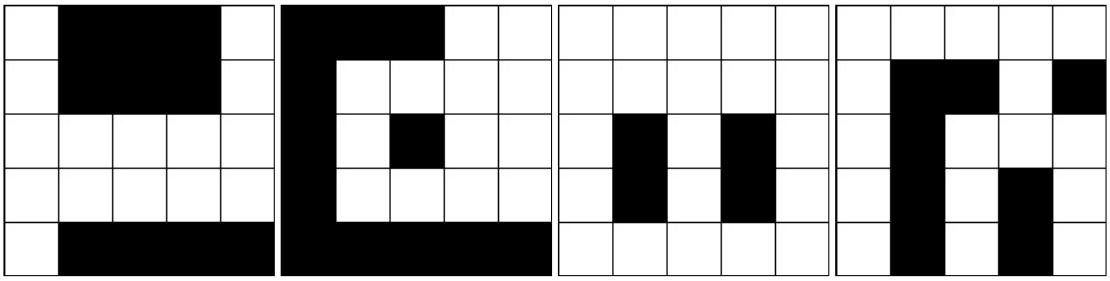
## 3.3 Experiment Results

We implemented the HHPOMDP algorithm in Python and built a simulation environment on grids in Robot Operating System (ROS). The program is running on Intel Xeon CPU of 2.30GHz with Ubuntu 16.04.4 LTS. The simulation system consists of an environment canvas node for displaying the map, a pursuer planning node, and a random target node. Firstly, we run our simulation based on the following four maps in shown Fig. 3.3. These four maps have different indoor structures. Map 2 has a loop and Map 3 has two loops, which would give the target more possibility to evade and hide. In these maps, catching the target is more difficult for the pursuers than those maps with dead-end as the target has more chances to get out from the line of sight of the pursuers.

Fig. 3.4 shows the number of nodes expanded in Map 1–4 with both the HHPOMDP and a standard POMDP. The graph is drawn in a log scale, thus our method significantly decreases the number of nodes expanded. The number of nodes for the standard POMDP method is also exponentially increasing as the number of robots grows. The standard full POMDP is solved using a standard value iteration as described in [10]. Solving the standard POMDP takes more than 48 hours with more than 2 robots in our simulation environment.

We run our method in Map 4 with multiple robots up to three. The result is shown in Fig. 3.5.

The number of nodes in base MDP is independent from the number of robots, but the number of nodes grows for both the abstract POMDP and transition states. Fig. 3.5b shows the time of capturing the target with different numbers of robots. Both the average and standard deviation of



(a) Map 1 (b) Map 2 (c) Map 3 (d) Map 4

Figure 3.3: Environment maps with different indoor structures

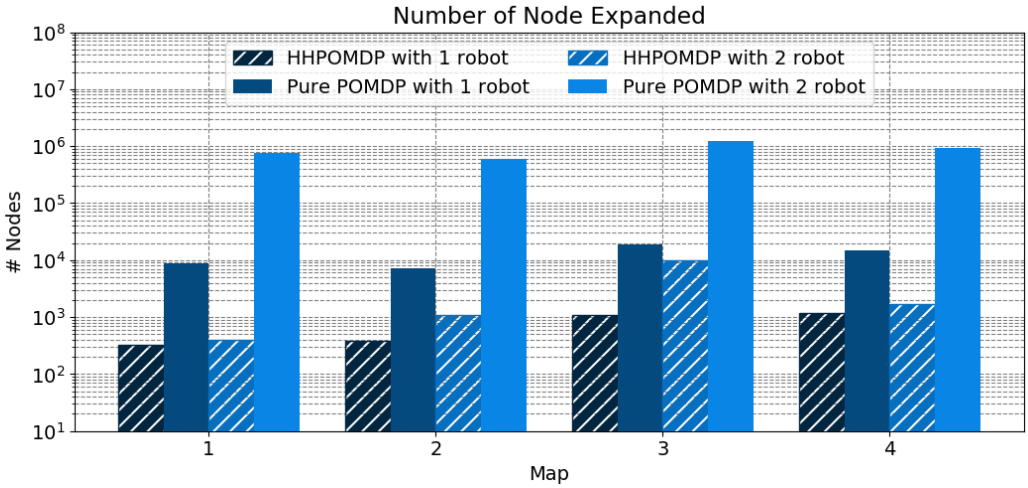
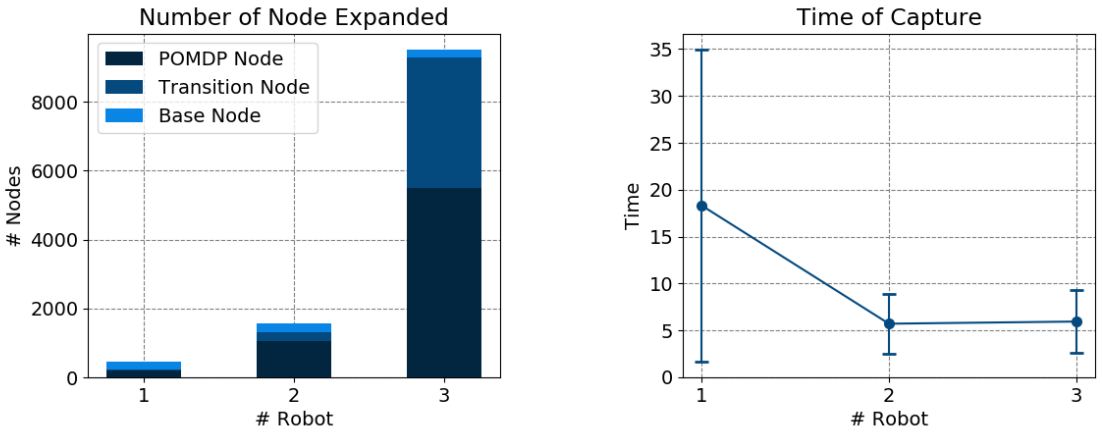
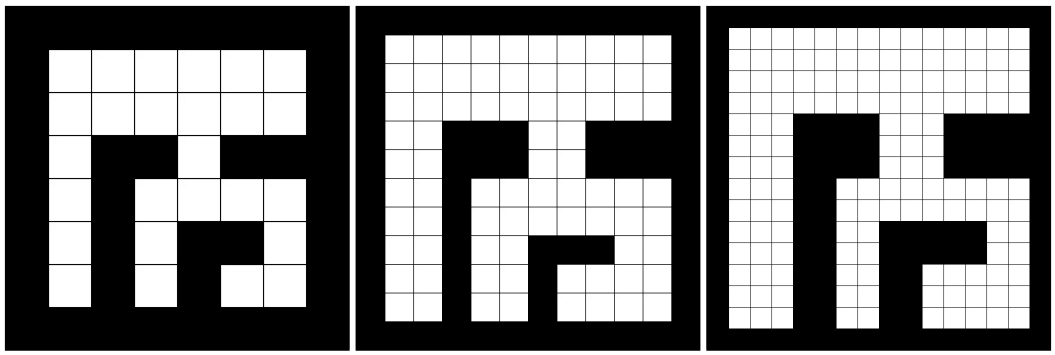



Figure 3.4: The number of nodes expanded in our HHPOMDP and a standard POMDP



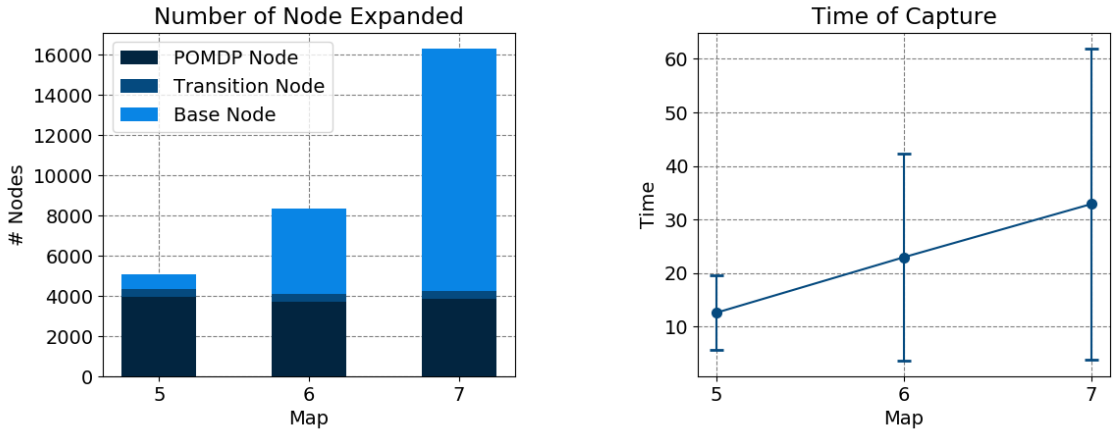

(a) (b)

Figure 3.5: Results with Map 4 with different numbers of robots



(a) Map 5 (b) Map 6 (c) Map 7

Figure 3.6: Environment maps with different sizes




(a) (b)

Figure 3.7: Result with 2 robots on maps of different size

the capture time decreases if the number of robots increases from one to two. It remains the same with three robots, mainly because the map only contains dead-ends but not loops, thus increasing the number of robots does not improve the performance of the robots in capturing the target.

We also tested our algorithm with different sizes of maps as shown in Fig. 3.6. They are of 8-by-8, 12-by-12, 16-by-16 grid maps with the same structure, i.e., convex hulls and borders. The result with two robots in the three different map sizes is shown in Fig. 3.7. Since the environment has the same structure, number of nodes for the abstract POMDP remains the same, only the number of the base MDP increases while the size of the map grows. Both the average and standard deviation for the time of capturing the target increases with the map size, since there is more free spaces for the target to move and thus it is more difficult to catch the target.

# Chapter 4

# Control with Specialization

## 4.1 Problem Definition

Consider a homogeneous robotic team with $n$ robots in an indoor environment, with positions denoted as $x_i \in \mathbb{R}^2$ where $i \in \{1, 2, \ldots, n\}$, and single integrator dynamics of $\dot{x}_i = u_i$. Velocity limitation exists for the robots as $||u_i|| \leq u_{max}$. Each robot is able to communicate with all robots within a limited Euclidean distance $R$, which means that robot $i$ is connected and can communicate with robot $j$ if $||x_i - x_j|| \leq R$, with $i, j \in \{1, \ldots, n\}$. In this case, robot $j$ is considered as a neighbor of robot $i$. We denote the neighbor set of robot $i$ to be $\mathcal{N}_i$. This forms a spacially induced connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each vertex $v_i \in \mathcal{V}$ represents robot $x_i$ in the environment, and each edge $e_{ij} \in \mathcal{E}$ between $v_i$ and $v_j$ exists when robot $i$ and robot $j$ is connected as defined above. With the definitions above, the connectivity graph $\mathcal{G}$ is undirected, i.e. $e_{ij} = e_{ji}$.

When the robots move, they maintain a safety distance with each other of $d_{ij} \geq r$ where $d_{ij} = ||x_i - x_j||$ and $r$ is the safety radius between the robot $i$ and $j$ to avoid collision. The robots also avoid obstacles in the environment by maintaining $||x_i - x_o|| \geq r$ where $x_o \in \mathcal{O}$ denotes the list of obstacles in the environment, which is known to the robots. To guarantee connectivity of the system as well as avoiding obstacles in the environment, we maintain the *Minimum Connectivity Constraint Spanning Tree* [18] with *barrier certificate* [3].

Suppose some robots are assigned tasks and need to visit a sequence of locations in parallel, and the other robots are moving to keep the system connected, i.e. keep the connectivity graph $\mathcal{G}$ strongly connected. Consider a task mapping function specified as $s[i, t] = g_i$, where $g_i \in \mathbb{R}^2$ denotes the goal location of robot $i$ at time $t$. We define those robots with assigned tasks to be *task robots*. If robot $i$ does not have any assigned task at time $t$, i.e. $s[i, t] = \varnothing$, then robot $i$ is considered as a *connection robot*, whose role is to maintain the connectivity of the robotic system and provide high flexibility for task robots to perform assigned tasks. We denote the set of connection robots to be $\mathcal{V}_c$ and the set of task robots to be $\mathcal{V}_t$. According to definition, we have $\mathcal{V}_c \cup \mathcal{V}_t = \mathcal{V}$. Since the robots are homogeneous, they can be either task robot or connection robot depending on task allocations and may switch roles accordingly.

An illustrative example is shown in Figure 4.1 to demonstrate the possible functionality of the connection robots in an indoor environment. As shown in the figure, the red robots are the

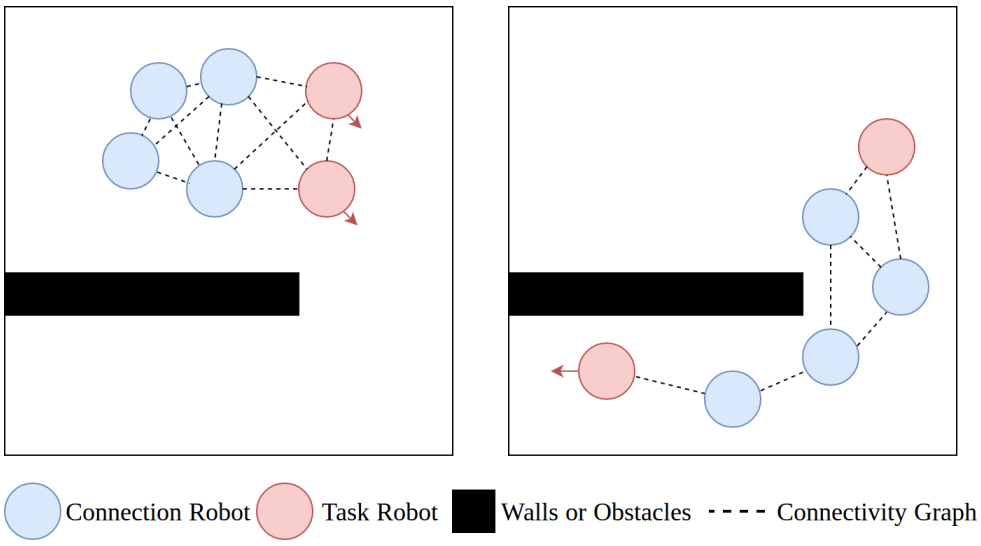



Figure 4.1: Two example scenarios for a system with robots allocated with tasks and connection robots. Left: connection robots need to keep up with task robots to avoid being left behind; Right: connection robots need to spread out so as to provide flexibility for the task robots to execute tasks.

task robots and the blue robots are those without tasks and thus serving as connection robots to keep the system connected while others executing their tasks. The connection robots need to keep up with the task robots when the goal locations of the task robots are in a similar direction. In another example scenario when the goal locations of the task robots are far away from each other, the connection robots need to adjust the topology of the connectivity graph so as to provide maximum flexibility for the task robots to achieve their goal locations. The walls and task locations heavily influence the topology of the connectivity graph, thus having a huge impact on the overall performance.

As described in the example above, depending on the environment and task locations, the topology of the connectivity might change and could result in failing assigned tasks. In this case, it is essential for the connection robots to correct the unsuitable topology of the connectivity graph for better support of the task executions. Therefore, in this paper, our objective is to design a distributed controller for the connection robots to correct topology of the connectivity graph so as to maintain flexible connectivity of the robotic system.

## 4.2 Topology Correction

### 4.2.1 Graph Laplacian and Convergence

Consider the *graph Laplacian* matrix $L$ of the above mentioned connectivity graph $\mathcal{G}$. The graph Laplacian is defined as

$$L = D - A \tag{4.1}$$

where $A$ is the adjacency matrix where each element $a_{ij} = 1$ if an edge exists between $v_i$ and $v_j$, and zero otherwise. $D = \text{diag}(deg(1), \ldots, deg(n))$ is the degree matrix where each $deg(i)$

denotes the degree of vertex $v_i$ that $deg(i) = \sum_{i \neq j} a_{ij}$, and zero off-diagonal elements. By definition, $L$ is symmetric if the graph is undirected, and has a right eigen vector of $\mathbf{1}$ and zero eigenvalue, i.e. $L \cdot \mathbf{1} = 0 \cdot \mathbf{1}$.

Laplacian matrix $L$ of the connectivity graph is essential for evaluating the convergence of consensus algorithms. The eigenvalues of $L$ can be computed and sorted in ascending order, denoted as

$$\lambda_1(L) \leq \lambda_2(L) \leq \ldots \leq \lambda_n(L) \tag{4.2}$$

The smallest eigenvalue $\lambda_1(L)$ of the Laplacian matrix is always zero as mentioned above. The second smallest eigenvalue $\lambda_2(L)$ describes the *algebraic connectivity* of the graph [7]. Let $\delta = \min deg(i)$ to be the minimum degree of the vertices in graph $\mathcal{G}$ with $n$ vertices. A lower bound for the algebraic connectivity $\lambda_2(L)$ exists [5]

$$\lambda_2(L) \geq 2\delta - n + 2 \tag{4.3}$$

It is known that a continuous-time consensus is globally exponentially reached with a speed that is faster or equal to $\lambda_2(L_s)$ where $L_s = (L + L^T)/2$ for a strongly connected balanced digraph [24]. For undirected graph with a symmetric Laplacian matrix, we have $L = L_s = (L + L^T)/2$. In our setting, the speed of convergence is $\lambda_2(L)$.

Therefore, to achieve a faster convergence rate, the controller of the connection robots aims at correcting the topology of connectivity graph by *maximizing the minimum degree* $\delta$.

### 4.2.2 Weighted Rendezvous

To keep the robotics system connected, it is straight forward to execute *rendezvous* on the connection robots so that they could keep up with the task robots. A common control law used for rendezvous [24] is

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j(t) - x_i(t)) \tag{4.4}$$

where $\mathcal{N}_i$ denotes the neighbor set of robot $i$ as introduced in section 4.1 and $a_{ij}$ is the element in adjacency matrix. However, this pure rendezvous controller will result in clusters of robots, making it hard for the non-connection robots to execute tasks. Following the discussion in section 4.2.1, we propose the *weighted rendezvous* as follows

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} w^r_{ij}(x_j(t) - x_i(t)) \tag{4.5}$$

where the weight $w^r_{ij}$ in the weight array $\mathbf{w}^r_i = \left[w^r_{i1}, \ldots, w^r_{ij}, \ldots, w^r_{i|\mathcal{N}_i|}\right]$, where $j \in \mathcal{N}_i$.

#### Degree-based Weighted Rendezvous

Following the discussion in section 4.2.1, we may set the weights with respect to the degree of neighboring vertices. The weights should be larger on vertices with smaller degrees and smaller on vertices with larger degrees. Therefore, the desired weights are calculated as

$$\mathbf{w}^r_i = \max deg(\mathcal{N}_i) - deg(\mathcal{N}_i) + \epsilon \tag{4.6}$$

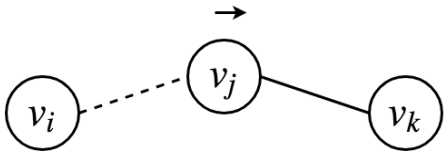



Figure 4.2: The moment when vertex $v_j$ is moving to $v_k$ resulting in $v_i$ and $v_j$ getting disconnected from each other.

where $deg(\mathcal{N}_i)$ denotes the degree array of the neighboring vertices where each element is the degree of a neighbor vertex $deg(j)$, $j \in \mathcal{N}_i$ and $|deg(\mathcal{N}_i)| = |\mathcal{N}_i|$. The variable $\epsilon$ is a small value of $1 \gg \epsilon > 0$ and serves as a *correction factor* to guarantee that weights are not all zeros with complete graph (the graph that every pair of vertices is connected). The weight array is then normalized so that $\sum_{j \in \mathcal{N}_i} w_{ij}^r = 1$.

We will then prove that this controller will correct the topology of connectivity graph by modifying the minimum degree $\delta$ of graph $\mathcal{G}$.

An assumption made in this paper is that time discretization is dense enough so that only one edge will change (connect or disconnect) at each time step. Before we start the proof, we list several cases of topology changes that we are not interested in:

- Adding an edge: when two robots come closer and new connection is made, it can only cause an increase of minimum degree $\delta$.
- Removing an edge $e_{ij}$ where $deg(i) > \delta$ and $deg(j) > \delta$: when two robots $i$ and $j$ move away from each other with a distance larger than $R$, the connection edge $e_{ij}$ is removed. Let $deg(i,t)$ denotes the degree of vertex $v_i$ at time $t$. When originally at time $t$, $deg(i,t) > \delta$, notice that degrees are integers, we have $deg(i,t) \geq \delta + 1$. Thus, if $e_{ij}$ is removed in the next time step, we still have $deg(i, t+1) \geq \delta$, similar inequality holds for $v_j$. The minimum degree $\delta$ is not influenced in the case.

Therefore, the only case that is of our interest is shown in Figure 4.2, which describes the case when the vertex $v_i$ having minimum degree $deg(i,t) = \delta$, and $v_j$ is moving towards its other neighbor $v_k$, causing edge $e_{ij}$ to disconnect. This is the only case that will cause a decrease of $\delta$ and in the following proof of theorem 1, the following discussion focuses on this case.

**Preposition 1.** *With the control law defined in equation 4.5 with weights defined in equation 4.6, the minimum degree $\delta$ of connectivity graph increases, i.e. $\delta(t+1) \geq \delta(t)$, where $\delta(t)$ denotes the minimum degree at time $t$.*

*Proof.* We prove by contradiction.

As described above, the only case that will result in $\delta(t) > \delta(t+1)$ is shown in Figure 4.2 when vertex $v_j$ moves towards $v_k$ and disconnects with vertex $v_i$ where $deg(i,t) = \delta(t)$ at time $t$. According to equation 4.5, $\dot{x}_j(t)$ moves towards $x_k(t)$ results from $w_{jk}^r > w_{ji}^r$. Then we have

$$\begin{aligned} w_{jk}^r &= \max deg(\mathcal{N}_j) - deg(k) + \epsilon \\ &> \max deg(\mathcal{N}_j) - deg(i) + \epsilon = w_{ji}^r \end{aligned}$$

Thus we have

$$deg(i) = \delta(t) > deg(k) \tag{4.7}$$

This violates the initial assumption that $v_i$ with $deg(i,t) = \delta(t)$ at time $t$ has the minimum degree. Therefore, the case described above will not happen and any other scenarios will not result in decrease of $\delta$, thus $\delta(t+1) \geq \delta(t)$. □

**Distance-based Weighted Rendezvous**

The degree-based approach introduced in the previous section is able to trigger edge changes so as to increase the convergence rate by gradually modifying the minimum degree $\delta$ in the graph. We notice that this degree-based approach treats all graphs with the same vertex and edge set the same, without considering the actual physical distance between vertices. To further optimize our objective, we propose the *distance-based weighted rendezvous* based on the degree-based approach that provides the same guarantees and better performance.

We first define a *density score* of each vertex $v_i$ as

$$c(i) = deg(i) + \sum_{j \in \mathcal{N}_i} \frac{R - d_{ij}}{deg(i)(R-r)} \tag{4.8}$$

The density score $c(i)$ describes the level of density around vertex $v_i$. An intuition on this is that when the neighboring robots are close to robot $i$, $d_{ij}$ is relatively small, then density score of robot $i$ is higher, and vice versa. Similarly, the weights are calculated as

$$\mathbf{w}_i^r = \max c(\mathcal{N}_i) - c(\mathcal{N}_i) + \epsilon \tag{4.9}$$

where $1 \gg \epsilon > 0$ is the same *correction factor* as in previous section. The weight array is then normalized such that $\sum_{j \in \mathcal{N}_i} w_{ij}^r = 1$ for each robot $i$. We will then prove that this formulation gives the same guarantee as equation 4.6.

**Lemma 2.** *For any two connected vertices $v_i, v_j \in \mathcal{V}$, $e_{ij} \in \mathcal{E}$ on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, if $deg(i) > deg(j)$, then $c(i) \geq c(j)$.*

*Proof.* We prove by analyzing the relationship of $deg(i)$ and $c(i)$ for vertex $v_i$.

Since $d_{ij} = ||x_i - x_j||$ when $e_{ij} \in \mathcal{E}$, we have

$$r \leq d_{ij} \leq R \tag{4.10}$$

where as defined in section 4.1 that $r$ is the safety radius and $R$ is the connectivity radius. By applying the inequality in equation 4.10, we have

$$\begin{aligned} c(i) &= deg(i) + \sum_{j \in \mathcal{N}_i} \frac{R - d_{ij}}{deg(i)(R-r)} \\ &\geq deg(i) + \sum_{j \in \mathcal{N}_i} \frac{R - R}{deg(i)(R-r)} = deg(i) \end{aligned}$$

Notice that $deg(i) = |\mathcal{N}_i|$, we also have

$$\begin{aligned} c(i) &= deg(i) + \sum_{j \in \mathcal{N}_i} \frac{R - d_{ij}}{deg(i)(R-r)} \\ &\leq deg(i) + \sum_{j \in \mathcal{N}_i} \frac{R - r}{deg(i)(R-r)} = deg(i) + 1 \end{aligned}$$

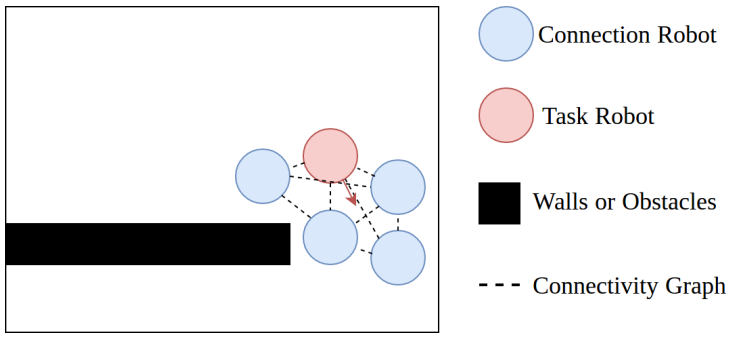



Figure 4.3: An example when the task robot is trying to move to an assigned location but the connection robots are cluttered along the way.

Combining the two equations above we get

$$deg(i) \leq c(i) \leq deg(i) + 1 \tag{4.11}$$

Notice that degree of vertex is integer. The inequality $deg(i) > deg(j)$ is the same as $deg(i) \geq deg(j) + 1$. Therefore we have

$$c(i) \geq deg(i) \geq deg(j) + 1 \geq c(j) \tag{4.12}$$

Then we may conclude that $c(i) \geq c(j)$ holds when $deg(i) > deg(j)$. □

We are then able to show our final statement. It is proven in Lemma 2 that density score preserves the inequality of the degree values. Following the same procedure of the proof for Prepostion 1, we may conclude that $\delta$ increases similarly for the density score defined in equation 4.8. Therefore, the following theorem holds.

**Theorem 3.** *With the control law defined in equation 4.5 with weights defined in equation 4.9, the minimum degree $\delta$ of connectivity graph increases, i.e. $\delta(t+1) \geq \delta(t)$, where $\delta(t)$ denotes the minimum degree at time $t$.*

With this distance-based weighted rendezvous, the connection robots will be able to keep up with the task robots and provide a flexible topology for the whole system.

### 4.2.3 Weighted Flocking

While weight rendezvous may solve the problem when task robots are moving away from the cluster of connection robots, weighted flocking introduced in this section may solve the problems caused by the scenario when task robots are moving through a cluster of connection robots. When there are many connection robots within the system, it is possible that they block the way of the task robots. An example of this is shown in Figure 4.3 when the task robot tries to move through a cluster of connection robots.

To solve the problem mentioned above, we propose *weighted flocking* as follows

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} w_{ij}^f u_j \tag{4.13}$$

The weights are designed so that the connection robots flocks with the neighboring task robots.

$$w_{ij}^f = \begin{cases} \frac{1}{|\mathcal{V}_t \cap \mathcal{N}_i|}, & \text{if } v_j \in \mathcal{V}_t \\ 0, & \text{otherwise} \end{cases} \tag{4.14}$$

where $|\mathcal{V}_t \cap \mathcal{N}_i|$ is the number of task robots in the neighbors of robot $i$.

### 4.2.4 Weighted Behavior Combination

The final resulting controller of the connection robots is designed as a *weighted behavior combination* that combines the output from both weighted rendezvous and weighted flocking.

Denote the output velocity with weighted rendezvous as $u_i^r$ and the output velocity with weighted flocking as $u_i^f$ for robot $i$. We then calculate the weights of performing weighted flocking for each robot

$$\alpha_i = \frac{1}{2u_{max}} \left|\left| u_i^r - \frac{1}{|\mathcal{V}_t \cap \mathcal{N}_i|} \sum_{j \in \mathcal{V}_t \cap \mathcal{N}_i} u_j \right|\right| \cdot \gamma \tag{4.15}$$

where $\gamma \in (0, 1]$ is an environment-dependent value that controls the range of $\alpha_i$ since $\alpha_i \in (0, \gamma]$. $\gamma$ should be smaller with a very cluttered environment, and could be larger in an environment with relative sparse obstacles.

The controller of weighted behavior mixing for connection robots is then designed as

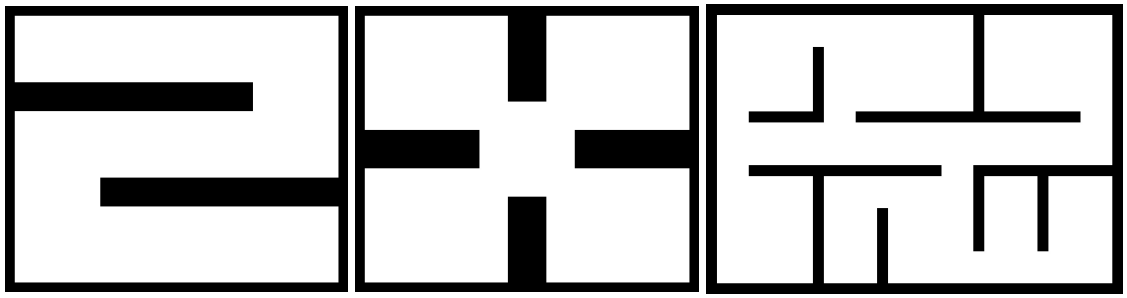
$$\dot{x}_i = (1 - \alpha_i) u_i^r + \alpha_i u_i^f \tag{4.16}$$

This method is also fully decentralized. Note that for weighted rendezvous and weighted flocking, each connection robot only needs to know the neighbor states to calculate the degree and density score, and then report to its neighbors. This distributed consensus-based method is very scalable with an increasing number of robots in the system, which we will show in the next section.

## 4.3 Results

We tested our controller with the maps in Figure 4.4. In map 4.4a, the task robots have a sequence of similar goal locations. The challenge for the connection robots in this setting is to avoid blocking the way for the task robots. In map 4.4b, the goal locations for the task robots are relatively far from each other in each step. The challenge here is that the connection robots need to stretch out so as to keep the task robots far away connected.

We tested and compared the performance of 1) weighted behavior combination, 2) no connection controller, 3) weighted rendezvous, 4) weighted flocking, and 5) rendezvous, in the maps shown in Figure 4.4. In 4.4a and 4.4b, he robotics system is given four tasks and the first three require two robots and the last one require one. The task locations in map 4.4a is relatively close and tasks in map 4.4b are far away. Due to space limitation, we will only show the results of two



(a) (b) (c)

Figure 4.4: (a) size 30×36, (b) size 30×36, and (c) size 27×39, are the three maps for the experiments. The black area is the walls or obstacles in the environment, and the white area is the free space.

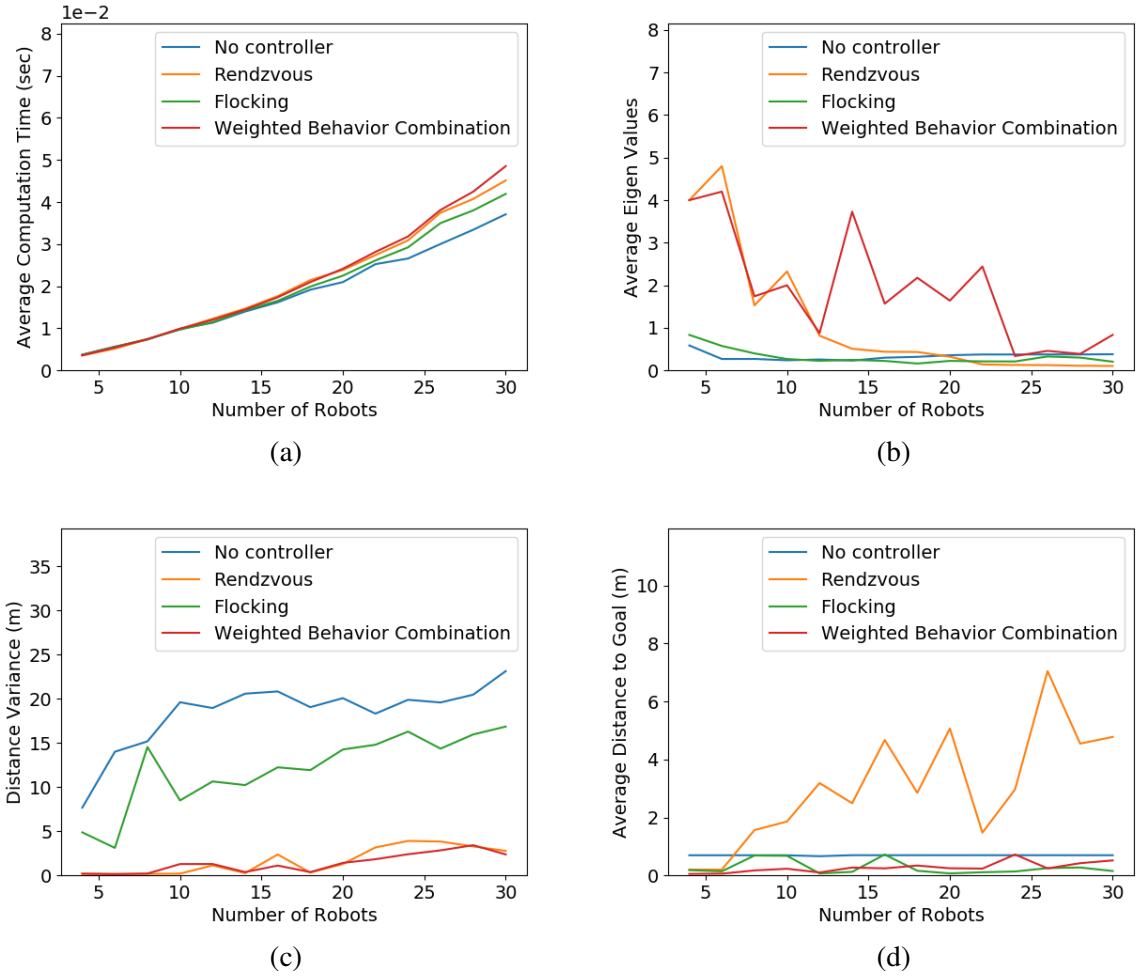
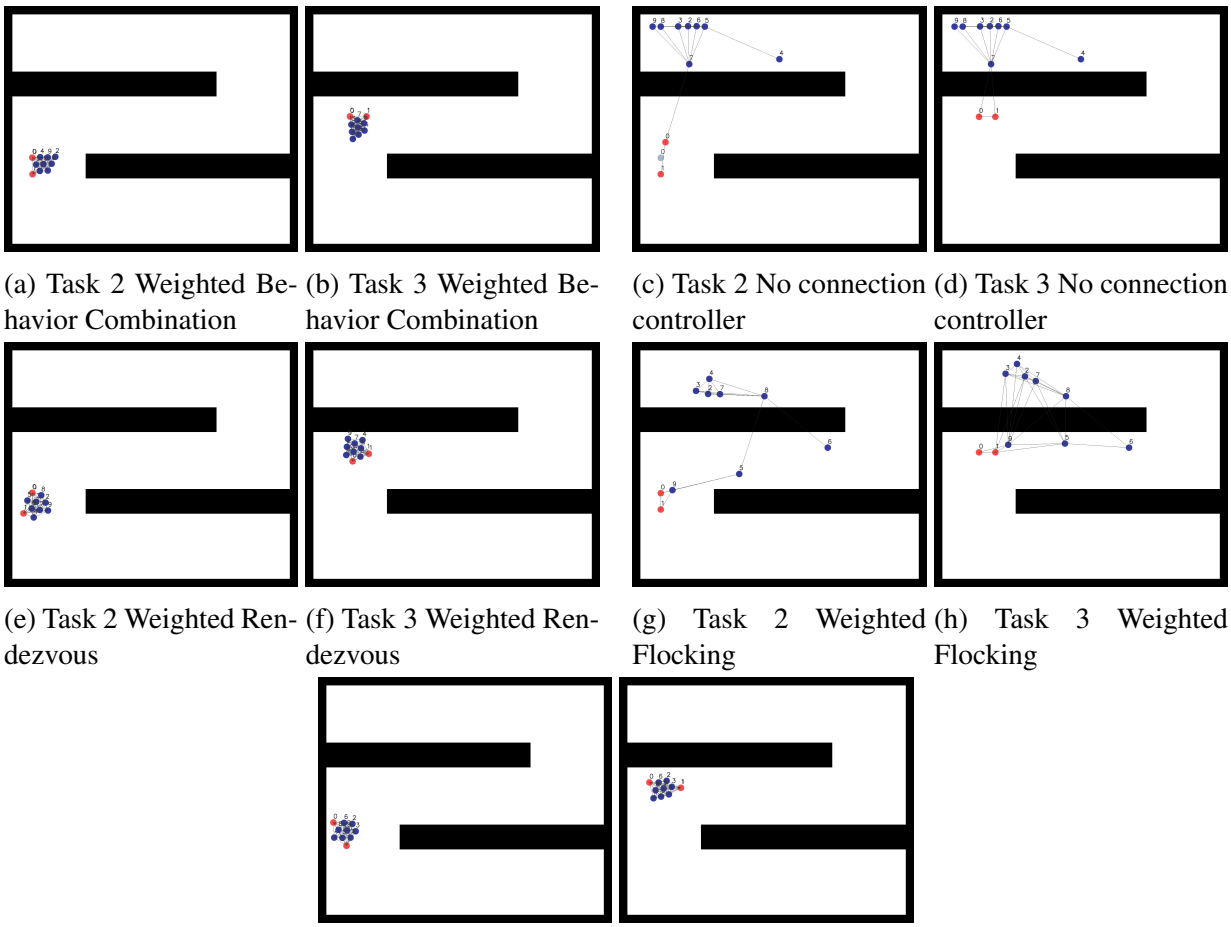



Figure 4.5: (a) Average computation time for each iteration; (b) Average eigenvalues (c) Variance of the robot locations at each time of convergence; (d) Average distance to goal for the task robots at each time of convergence

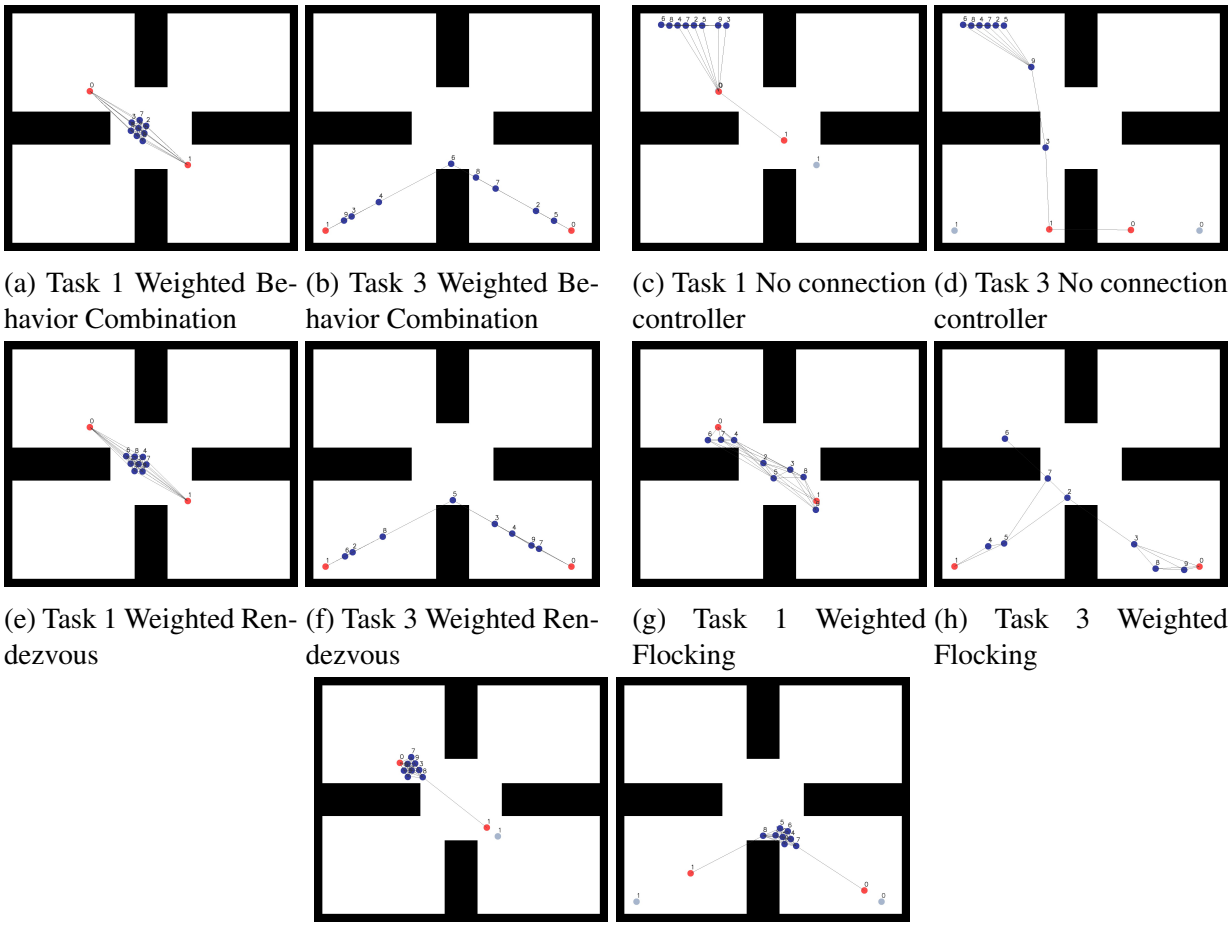


(a) Task 2 Weighted Behavior Combination (b) Task 3 Weighted Behavior Combination (c) Task 2 No connection controller (d) Task 3 No connection controller

(e) Task 2 Weighted Rendezvous (f) Task 3 Weighted Rendezvous (g) Task 2 Weighted Flocking (h) Task 3 Weighted Flocking

(i) Task 2 Rendezvous (j) Task 3 Rendezvous

Figure 4.6: Red robots: task robots, blue robots: connection robots, light blue circles: goal locations. The process of $N = 10$ robots executing task 2 and 3 in Map 4.4a with no connection controller. (a)-(b) Weighted behavior combination: the task robots are able to reach all the goal locations and the connection robots are able to keep up with the task robots till the end; (c)-(d) No connection controller: robot failing the task due to connectivity constraints; (e)-(f) Weighted rendezvous: connection robots blocking the task robots; (g)-(h) Weighted flocking: some robots were left behind; (i)-(j) Rendezvous: connection robots blocking the task robots.



(a) Task 1 Weighted Behavior Combination

(b) Task 3 Weighted Behavior Combination

(c) Task 1 No connection controller

(d) Task 3 No connection controller

(e) Task 1 Weighted Rendezvous

(f) Task 3 Weighted Rendezvous

(g) Task 1 Weighted Flocking

(h) Task 3 Weighted Flocking

(i) Task 1 Rendezvous

(j) Task 3 Rendezvous

Figure 4.7: Red robots: task robots, blue robots: connection robots, light blue circles: goal locations. The process of $N = 10$ robots executing task 1 and 3 in Map 4.4b with no connection controller. (a)-(b) Weighted behavior combination: the task robots are able to reach all the goal locations and the connection robots are able to keep up with the task robots till the end; (c)-(d) No connection controller: robot failing the task due to connectivity constraints; (e)-(f) Weighted rendezvous: perform the same as in weighted behavior combination; (g)-(h) Weighted flocking: some robots were left behind; (i)-(j) Rendezvous: robot failing the task due to connectivity constraints.

tasks on map 4.4a and map 4.4b. Other results will be shown in the video submitted. The results on Map 4.4a and Map 4.4b is shown in Figure 4.6 and Figure 4.7 respectively. In general, with weighted behavior combination, the task robots are able to reach all the goal locations and the connection robots are able to keep up with the task robots till the end. Rendezvous or weighted rendezvous may be too cluttered around each other and block the way of task robots resulting in failing the tasks. No connection controller or weighted flocking usually left some robots behind resulting in the task robots failing the tasks because of connectivity constraints.

We also measure the quantitative results of our experiments. The experiments include 10 runs with random initial locations and of a various number of robots in the system, ranging from $N = 4$ to $N = 30$. The method is written in python and test on Intel Xeon CPU E5-2660 with cores of 2.60GHz. We present our result on average computation time in Figure 4.5a, average eigenvalues representing convergence rate in Figure 4.5b, variance of distance in Figure 4.5c and average distance to goal locations in Figure 4.5d at the time of convergence. We may see that the computation time is approximately linear with the increasing number of robots in the system, thus it is scalable to a large number of robots. Also, the computation time for a system of $N = 30$ robots is only $0.05$ second on average. This shows that our approach can be run in real-time at each iteration. As shown in Figure 4.5b, our final method also outperforms other methods in terms of eigenvalues, with larger eigenvalues guarantee faster convergence rate. In terms of the overall performance, we take into account the variance in locations of the robots after convergence. This is used to measure the ability for the connection robots to keep up with the task robots so as to not be left behind. As shown in Figure 4.5c, our final method gives the smallest variance, which means that the connection robots are able to keep up with the task robot without being left behind. We also compute the average distance to goal locations for the task robots to measure the performance of completing assigned tasks for the task robots. As shown in Figure 4.5d, our final method is able to give a result as good as the best one. Above all, we may conclude that our weighted behavior combination outperforms all the other methods.

# Chapter 5

# Conclusion and Future Work

In this thesis, I proposed two approaches to handle scalability issues for multi-robot systems. Creating abstraction over state space in planning can save computation. Introducing connection robots in systems with a large number of robots could improve the overall performance.

First, I presented a scalable algorithm for an indoor pursuit-evasion problem using multiple robotic pursuers. I proposed the Hybrid Hierarchical POMDP structure that utilizes the convex hulls of the environment to create abstract states. The algorithm could transit between the base MDP and the abstract POMDP so as to keep track of the target when it disappears during the capture event. We have shown that our algorithm is more scalable than a standard multi-agent POMDP solution and can capture the target within a reasonable time.

The HHPOMDP approach is not limited only to the pursuit-evasion problem discussed in the paper. For other planning problems where an environment abstraction could be made at the lower level, having the hierarchical model would help with reducing computational complexity. The architecture could also be extended to scenarios when the planning tasks could be divided into groups and each group is independent of each other.

Our idea can be extended in the future to consider limited communication. For our current settings, all the robots are fully connected, thus the belief space information is shared across all robots throughout all time stamps. However, in real-world applications, keeping all robots connected and exchanging information are both unsafe and limited owing to bandwidth restrictions. Therefore, an extension of limited communication, information sharing, and belief space updates will be essential to increase the applicability of this problem. Since our current method is fully centralized, a decentralized version of the algorithm might be needed for the limited communication setting. In addition, we only consider scenarios with one target. However, in many real-world scenarios such as search and rescue, there might be multiple moving targets. In this case, task allocation is needed and optimizing over multiple targets is challenging in larger state spaces. This would require more abstraction over both the environment and target locations.

Another limitation of the current structure is that HHPOMDP is solved by the forward exploration in the reachable belief space, which is not efficient with growing state space. A more efficient solver would be needed, possibly with heuristic or sampling in policy trees.

In the second part, we considered the problem of controlling *connection robot* to maintain flexible connectivity graph for the robots with an assigned controller to achieve their goal. We proposed the method of combining weighted rendezvous and weighted flocking for the connec-

tion robots to keep up with the task robots as well as providing flexibility for the task robots to reach their goal by correcting the topology of the current connectivity graph with a provably fast convergence rate, with a distributed fashion. Both theoretical analysis and experimental result are presented. Results have shown that our weighted behavior combination method outperforms all other methods in overall performance. Our algorithm is also scalable to a large number of robots in the system so that more complex tasks or behaviors could also be achieved.

Both algorithms improve the scalability of the multi-robot systems by reducing computation complexity with abstract state space and controlling with specialization. These benefit the planning and control of multi-robot systems and enables more complicated applications for the system in various scenarios.

# Bibliography


[1] Sachiyo Arai, Katia Sycara, and Terry R Payne. Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. In *Pacific Rim International Conference on Artificial Intelligence*, pages 125–135. Springer, 2000. 2.2

[2] Richard Bellman. *Dynamic programming*. Courier Corporation, 2013. 3.2.4

[3] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015. 2.3, 4.1

[4] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005. 1

[5] Nair Maria Maia De Abreu. Old and new results on algebraic connectivity of graphs. *Linear algebra and its applications*, 423(1):53–73, 2007. 4.2.1

[6] William F Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):398–403, 1977. 3.2.2

[7] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973. 2.3, 4.2.1

[8] Nakul Gopalan, Marie desJardins, Michael L Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson LS Wong. Planning with abstract markov decision processes. In *International Conference on Automated Planning and Scheduling*, 2017. 2.2

[9] Leonidas J Guibas, Jean-Claude Latombe, Steven M LaValle, David Lin, and Rajeev Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry & Applications*, 9(04):471–493, 1999. 2.1

[10] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000. 3.2.1, 3.2.4, 3.2.4, 3.3

[11] Geoffrey Hollinger and Sanjiv Singh. Multi-robot coordination with periodic connectivity. In *2010 IEEE International Conference on Robotics and Automation*, pages 4457–4462. IEEE, 2010. 1

[12] Geoffrey Hollinger, Athanasios Kehagias, and Sanjiv Singh. Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *IEEE International Conference on Robotics and Automation*, pages 3870–3876. IEEE, 2007. 2.1

[13] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient

multi-robot search for a moving target. *The International Journal of Robotics Research*, 28 (2):201–219, 2009. 1, 2.1, 2.2

[14] M Ani Hsieh, Anthony Cowley, Vijay Kumar, and Camillo J Taylor. Maintaining network connectivity and performance in robot teams. *Journal of field robotics*, 25(1-2):111–131, 2008. 1

[15] Volkan Isler, Dengfeng Sun, and Shankar Sastry. Roadmap based pursuit-evasion and collision avoidance. In *Robotics: Science and Systems*, volume 1, pages 257–264, 2005. 2.1

[16] George Kantor, Sanjiv Singh, Ronald Peterson, Daniela Rus, Aveek Das, Vijay Kumar, Guilherme Pereira, and John Spletzer. Distributed search and rescue with robot and sensor teams. In *Field and Service Robotics*, pages 529–538. Springer, 2003. 1

[17] Wenhao Luo and Katia Sycara. Adaptive sampling and online learning in multi-robot sensor coverage with mixture of gaussian processes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6359–6364. IEEE, 2018. 1

[18] Wenhao Luo, Sha Yi, and Katia Sycara. Behavior mixing with minimum connectivity maintenance for large-scale multi-robot systems. *Paper submitted*. 4.1

[19] Zhenqiang Mi, Yang Yang, and Guangjun Liu. Hero: A hybrid connectivity restoration framework for mobile multi-agent networks. In *2011 IEEE International Conference on Robotics and Automation*, pages 1702–1707. IEEE, 2011. 1

[20] Nathan Michael, Michael M Zavlanos, Vijay Kumar, and George J Pappas. Maintaining connectivity in mobile robot networks. In *Experimental Robotics*, pages 117–126. Springer, 2009. 2.3

[21] Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on automatic control*, 50(2):169–182, 2005. 1

[22] Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. Automated sequencing of swarm behaviors for supervisory control of robotic swarms. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2674–2681. IEEE, 2017. 1, 2.3

[23] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006. 2.3

[24] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007. 1, 2.3, 4.2.1, 4.2.2

[25] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010. 2.2

[26] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. 3.2.3

[27] David V Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002. 3.2.1

[28] 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. 1, 2.3

[29] Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008. 1, 2.3

[30] 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