Indoor Pursuit-Evasion with Hybrid Hierarchical Partially Observable Markov Decision Processes for Multi-Robot Systems

Sha Yi*, Changjoo Nam, and Katia Sycara

Carnegie Mellon University

Abstract. In this paper, we examine a pursuit-evasion problem where more than one pursuer may search for one evader in indoor environments. Partially Observable Markov Decision Processes (POMDPs) provide a framework to model the uncertainty arisen from the unknown location of the evader. However, the approach is intractable even with a single pursuer and an evader. Therefore, we propose a Hybrid Hierarchical POMDP structure for improved scalability and efficiency. The structure consists of (i) the base MDPs for the cases where the evader is visible to the pursuers, (ii) the abstract POMDPs for the evader states that are not directly observable, and (iii) the transition states bridging between the base MDPs and abstract POMDPs. This hybrid approach significantly reduces the number of states expanded in the policy tree to solve the problem by abstracting environment structures. Experimental results show that our method expands only 5% of nodes generated from a standard POMDP solution.

Keywords: Pursuit and Evasion, Multi-Robot Systems, Markov Decision Processes

1 Introduction

The pursuit-evasion problem where more than one robots may pursue a non-adversarial evader has long been considered as an important problem because of its wide applicability to many real world scenarios. For example, a team of robots may search an area to locate a target, which could be their teammate which has lost its connection with the team or a victim of a disaster. Since the movement of the target is not known to the team, locating the target quickly is not an easy task to the team without an efficient search strategy.

In this paper, we consider the pursuit-evasion problem in indoor environments which have received significant attention owing to the complex structure which makes the problem more difficult. We also consider multiple pursuers as utilizing

*All authors are with the Robotics Institute at Carnegie Mellon University. E-mail: shayi@andrew.cmu.edu

This research was funded by AFOSR awards, FA9550-18-1-0097 and FA9550-15-1-0442.

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 early work [8] 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 (HHPOMDPs)*. Our framework creates an abstraction of the environment and reduces the state space for improved scalibility. 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 pursuer 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, present the detailed algorithm in Sec. 4, and finally present experimental results and comparisons in Sec. 5.

2 Related Work

Pursuit-evasion has received much attention for decades. Guibas [5] 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. [9] 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. [7, 8] 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.

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 [8] 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 [4] introduced abstraction to the state space for top-down policy search for MDP. However, the joint space of POMDPs is still intractably large. Ong [10] 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.

3 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 ith 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:

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

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:

$$\pi^\star = \arg\max_{\pi} V(s). \tag{2}$$

4 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.

4.1 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 [6]. The process is described by $(S, A, \Theta, T, O, R, \gamma)$, where S is the state space, A is the action set, Θ 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 γ is the discount factor. The goal

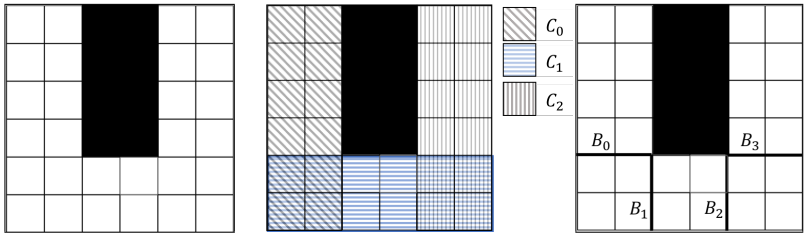
is to compute an optimal policy $\pi^\star : S \times O \rightarrow A$ that maps from each state and observation pair to an action that maximize the expected discounted reward.

For multi-agent POMDP with k agents [12], 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 $\pi_i^\star : S_i \times O_i \rightarrow A_i$ maps from each state and observation pairs to a corresponding action that maximize the expected discounted reward.

4.2 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.



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

Fig. 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.

As shown in Fig. 1, the grip map of the environment is modeled in the following way: Fig. 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. 1b, the convex hulls may have overlapping areas. The overlapping areas of the adjacent convex hulls form borders shown in Fig. 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 [3].

4.3 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. 4.1. 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 [11].

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. 2.

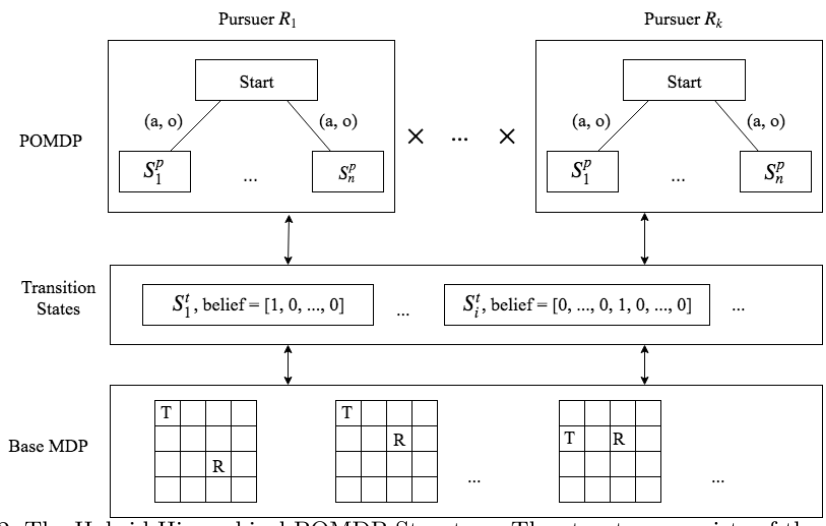


Fig. 2: The Hybrid Hierarchical POMDP Structure. The structure consists of the base MDPs for the cases where the evader is visible to the pursuers, the abstract POMDPs for the evader states that are not directly observable, and the transition states bridging between the base MDPs and abstract POMDPs.

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. 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 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.

4.4 Transitions and Rewards

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

$$V^*(s^b) = \max_{a^b \in A^b} \sum_{s^{b'}} T^b(s^b, a^b, s^{b'})[R(s^b, a^b, s^{b'}) + \gamma V^*(s^{b'})] \tag{3}$$

where s^b is the current base state in the grid world, which is the coordinate on the grid map, γ 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 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:

$$T^b(s^b, a^b, s^{b'}) = p(s^{b'}|s^b, a^b) = \frac{1}{|S_T^b|}. \tag{4}$$

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:

$$R(s^b, a^b, s^{b'}) = \begin{cases} r, \ \exists j \in [0, k) \ s.t. \ s^b_{R_j} = s^b_T \\ 0, \ \text{otherwise.} \end{cases} \tag{5}$$

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, Θ: 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
```

4.4.2 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.'

4.4.3 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 [6]

$$V^*(b) = \max_{a \in A} \left\{ \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{o \in \Theta} \sum_{s \in S} T(o|s,a)b(s)V^*(\tau(b,o,a)) \right\} \tag{6}$$

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 τ 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 $A = A^p_T \times \Pi_j A^p_{R_j}$. Similarly, the robot state space is also the Cartesian product $s^p_R = \Pi_j s^p_{R_j}$. 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

$$T(o|s,a) = T(o|s^{p\,\prime}_T, s^{p\,\prime}_R) \cdot T(s^{p\,\prime}_T|s^p_T) \cdot T(s^{p\,\prime}_R|s^p_R). \tag{7}$$

Then we have the following modified Bellman equation:

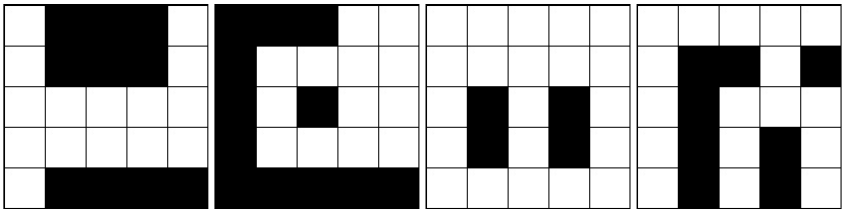
$$V^*(s^p) = \max_{a \in A} \left\{ \gamma \sum_{o \in \Theta} \sum_{s^{p\,\prime}_T} \sum_{s^{p\,\prime}_R} T(o|s^{p\,\prime}_T, s^{p\,\prime}_R) T(s^{p\,\prime}_T|s^p_T) T(s^{p\,\prime}_R|s^p_R) b(s^p_T) V(\tau(b,o,a)) \right\}. \tag{8}$$

The HHPOMDP system is solved by value iteration based on forward exploration from the root state [6], which will only search in all reachable belief states instead of the full span of belief vectors.

5 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. 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.



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

Fig. 3: Environment maps with different indoor structures

Fig. 4 shows the number of nodes expanded in Map 1–4 with both the HH-POMDP 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 [6]. Solving the standard POMDP takes more than 48 hours with more than 2 robots in our simulation environment.

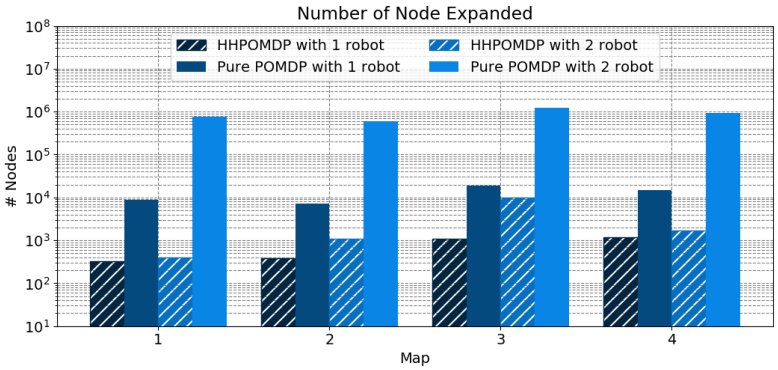
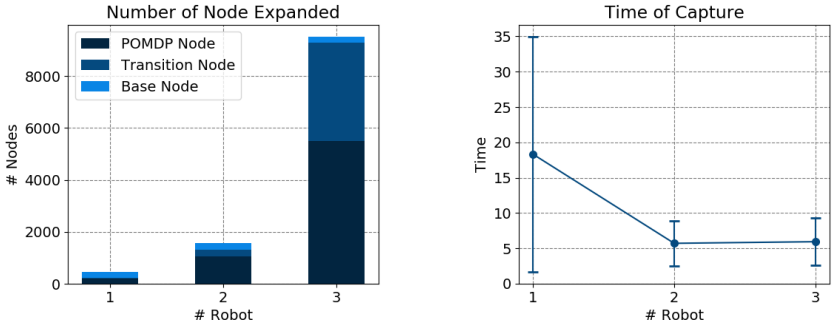


Fig. 4: The number of nodes expanded in our HHPOMDP and a standard POMDP

We run our method in Map 4 with multiple robots up to three. The result is shown in Fig. 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. 5b shows the time of capturing the target with different numbers of robots. Both the average and standard deviation of 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,

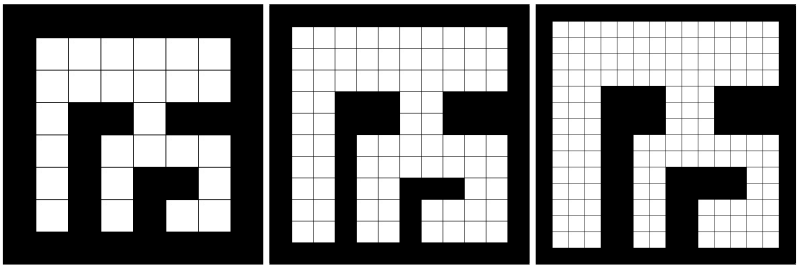


(a) (b)

Fig. 5: Results with Map 4 with different numbers of robots

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. 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. 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.

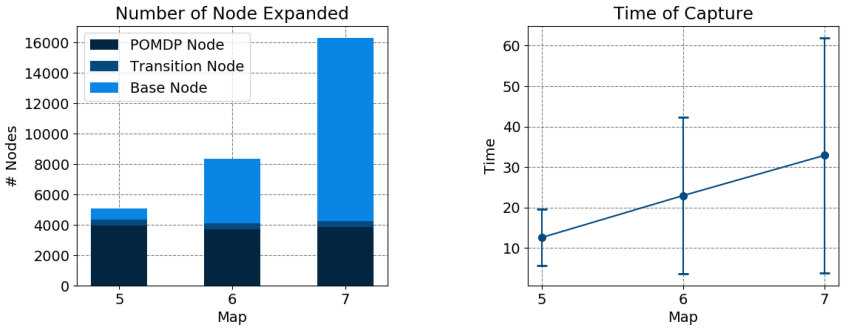


(a) Map 5 (b) Map 6 (c) Map 7

Fig. 6: Environment maps with different sizes

6 Conclusions and Future Work

This paper presents a scalable algorithm for an indoor pursuit evasion problem using multiple robotic pursuers. We 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 disappear during the capture



(a) (b)

Fig. 7: Result with 2 robots on maps of different size

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 in 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 from 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, extension on 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 target 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.

References

[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*. Springer. 2000, pp. 125–135.

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

[3] William F Eddy. "A new convex hull algorithm for planar sets". In: *ACM Transactions on Mathematical Software (TOMS)* 3.4 (1977), pp. 398–403.

[4] Nakul Gopalan et al. "Planning with abstract Markov decision processes". In: *International Conference on Automated Planning and Scheduling*. 2017.

[5] Leonidas J Guibas et al. "A visibility-based pursuit-evasion problem". In: *International Journal of Computational Geometry & Applications* 9.04 (1999), pp. 471–493.

[6] Milos Hauskrecht. "Value-function approximations for partially observable Markov decision processes". In: *Journal of Artificial Intelligence Research* 13 (2000), pp. 33–94.

[7] 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*. IEEE. 2007, pp. 3870–3876.

[8] Geoffrey Hollinger et al. "Efficient multi-robot search for a moving target". In: *The International Journal of Robotics Research* 28.2 (2009), pp. 201–219.

[9] Volkan Isler, Dengfeng Sun, and Shankar Sastry. "Roadmap Based Pursuit-Evasion and Collision Avoidance." In: *Robotics: Science and Systems*. Vol. 1. 2005, pp. 257–264.

[10] Sylvie CW Ong et al. "Planning under uncertainty for robotic tasks with mixed observability". In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1053–1068.

[11] Christos H Papadimitriou and John N Tsitsiklis. "The complexity of Markov decision processes". In: *Mathematics of Operations Research* 12.3 (1987), pp. 441–450.

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