Multi-Agent Patrolling under Uncertainty and Threats

Shaofei Chen[1,2], Feng Wu[2], Lincheng Shen[1], Jing Chen[1], and Sarvapali Ramchurn[2]

[1] College of Mechatronics and Automation, National University of Defense Technology, China
[2] School of Electronics and Computer Science, University of Southampton, UK

Abstract. We investigate a multi-agent patrolling problem in large stochastic environments where information is distributed alongside threats. The information and threat at each location are respectively modelled as a multi-state Markov chain, whose states are not observed until the location is visited by an agent. While agents obtain information at a location, they may suffer attacks from the threat at that location. The goal for the agents is to gather as much information as possible while mitigating the damage incurred. We formulate this problem as a Partially Observable Markov Decision Process (POMDP) and propose a computationally efficient algorithm to solve it. We empirically evaluate our algorithm in a simulated environment, and show that it outperforms a greedy algorithm up to 43% for 10 agents in a large graph.

Keywords: Multi-Agent Patrolling, Planning under Uncertainty, Partially Observable Markov Decision Process

1 Introduction

Multi-agent patrolling problems arise in many real-word applications, such as disaster response and military missions in hostile environments. Previous work has focused on how to guard the targets [1–6] or how to gather information as much as they can [7, 8]. But in many real scenarios, agents gather information while facing to a number of threats. Such threats may be natural ones (such as fire and radiation) or adversarial ones (such as terrorists). For example, when agents visit a building in a disaster area, the building states (intact, about to collapse, collapsing or collapsed) may correspond to threat states (levels) for agents, and the threat at each location may be changing stochastically, such that it switches between "about to collapse" to "collapsed" due to an aftershock [9]. When patrolling after an earthquake, a team of agents may share the threats observed at each position and estimate the damage they might suffer of visiting each position and revise the patrol schedule. In turn, in hostile scenarios, an adversary may introduce some threats in the environment and continuously change their distribution stochastically. For example, a group of unmanned aerial vehicles (UAVs) executing a reconnaissance mission in a specific area may face an adversary that may setup explosives near each target. As the attack resources may be limited, the adversary may schedule the distribution dynamically. The information in each environment may also change dynamically (e.g., a victim may get out of danger or the fire may get close to a victim).

While research in the area of single mobile agent path planning has considered threats as a probability and typically try to devise steps to avoid it at all cost [10, 11], they do not consider scenarios where the agents cannot simply avoid threats as above. Moreover, previous work on information gathering in dynamic environments [8] have focused on specific environmental phenomena (e.g., monitoring algal bloom growth in lakes) rather than stochastic events as in our scenarios. [3, 4] consider when an agent should stop observe in security game.

Against this background, we present a new online multi-agent patrolling algorithm under uncertainty and threats from the environment. The information and threat at each location are respectively modelled as a multi-state Markov chain, whose states are not observed until the location is visited by some agent. When visiting a location in the environment, agents acquire information corresponding to the information state and reset the information state variable. Moreover, agents may suffer a level of damage associated with the threat state (captured by Markov chain that remains unaffected by the actions of agents). The goal for the agents then to gather as much information as possible and minimize damage incurred. In more detail, we formulate the problem as a Partially Observable Markov Decision Process (POMDP) and introduce a belief state of reduced dimension for the problem. We then propose a predictive heuristic and develop an online single-agent algorithm. Instead of computing a joint policy for multiple agents, our multi-agent algorithm sequentially computes policies for individual agents. In particular, we extend the work of [7, 8] in our multi-agent algorithm to account for the cost of sequentially computing policies for individual agents. Thus, this paper advances the state of the art in the following ways:

– We propose the first algorithm for multi-agent patrolling under uncertainty and threats. We model the information and threat as Markov chains at each location in the environment and cast the problem as a Partially Observable Markov Decision Process (POMDP).
– We design a predictive heuristic to prune the search space considerably and provide an online single-agent algorithm. Moreover, we propose a multi-agent algorithm that sequentially computes policies for individual agents.
– We empirically show that for 10 agents in a large graph, our algorithm outperforms greedy by more than 43%.

The remainder of this paper is structured as follows. First we model the environment. We then formulate the decision problem as a POMDP and provide an algorithm for individual agents. Building upon this, we then present our multi-agent algorithm and evaluate it in a number of realistic scenarios.

2 The Physical Environment

We model the environment that the agents need to patrol as an undirected graph $G = (V, E)$. The set of spatial coordinates V is embedded in Euclidean space and edges E encode the movements that are possible between them. The number of the vertices is $N = |V|$. Time is denoted as the set of time steps $T = \{t_1, t_2, \ldots\}$.

We assume each vertex holds two states for information and threat. The information state variable indicates different levels of the new information generated at a given vertex since the last time it was visited by an agent. The threat state variable reflects how much damage the agent will suffer when visiting this vertex. The two state variables at each vertex change dynamically as independent, discrete-time multi-state Markov chains.

To clarify our setting, consider an agent that enters into a building on fire. In our setting, this is equivalent to the agent visiting a node in the graph. The fire level (threat state variable) and valuable information about victims and assets (information variable) changes over time. While exploring the building, the agent may acquire some information and suffer some damage due to the fire. We elaborate on the formal model of these states in the next section.

2.1 Markov Model of Threat and Information

Markov models are widely used to model dynamic states in stochastic environments. In this paper, we model the threats and information as dynamically changing Markov chains. At each time step, an agent selects one adjacent location to visit based on the estimated information value and the prior observation of threat states at each location. It then obtains a reward based on the value of the information, and suffers a loss which is associated with the threat state. After that, the information state of the visited vertex reset immediately.

The threat states for each vertex $R = (R_1, R_2, \dots, R_{K_1})$ indicate the threat levels of each vertex $v \in V$. The “damage” that an agent suffers when visiting a vertex is captured by the formulation $h : R \to \mathbb{R}^+$. We denote $L = [h(R_1), \dots, h(R_{K_1})]$ as the damage value vector, where $h(I_k)$ corresponds the damage value an agent will lose if the threat state is R_k and we assume $h(k)$ is increases monotonically with $k \in \{1, \dots, K_1\}$. For example, fire level at a vertex has 4 states $R = (R_1, R_2, R_3, R_4)$, which correspond to 4 level of damage $L = (0, 4, 6, 10)$. Threat states remain unaffected by actions of agents, evolve with time as a Markov chain model independently. We denote the matrix of transition probabilities for each two of the K_1 threat states as:

$$P_R = \begin{bmatrix} p_{R11} & \cdots & p_{R1K_1} \\ p_{R21} & \cdots & p_{R2K_1} \\ \vdots & \ddots & \vdots \\ p_{RK_11} & \cdots & p_{RK_1K_1} \end{bmatrix} = \begin{bmatrix} P_{R1} \\ P_{R2} \\ \vdots \\ P_{RK_1} \end{bmatrix} \tag{1}$$

Similar to threat states, information states for each location $I = (I_1, I_2, \dots, I_{K_2})$ correspond to K_2 stages of information values which agents could get when visiting a given location. The value of information is determined by the function $f : I \to \mathbb{R}^+$, and $f(k)$ increases monotonically with $k \in \{1, \dots, K_2\}$. We denote $F = [f(I_1), \dots, f(I_{K_2})]$ as the information value vector, where $f(I_k)$ corresponds the information value an agent could get if the information state is I_k. For example, information at a vertex has 4 states $I = (I_1, I_2, I_3, I_4)$, which correspond to 4 information values $L = (0, 2, 5, 10)$. While the threat state changes without any effect from the actions of agents, the information state at a given vertex will reset to I_1 when there is an agent visiting this vertex (I_1 is the information state which means no new information was generated after last visit), and then independently evolves as a K_2-state Markov chain model with a matrix of transition probabilities for each two of the K_2 information states as:

$$P_I = \begin{bmatrix} p_{I11} & \cdots & p_{I1K_2} \\ p_{I21} & \cdots & p_{I2K_2} \\ \vdots & \ddots & \vdots \\ p_{IK_21} & \cdots & p_{IK_2K_2} \end{bmatrix} = \begin{bmatrix} P_{I1} \\ P_{I2} \\ \vdots \\ P_{IK_2} \end{bmatrix} \tag{2}$$

In this paper, we assume the state transition probabilities of $\{p_{Iij}\}$ and $\{p_{Rij}\}$ are known.[3]Having defined the information and threat models, we now need to plan the sequential patrolling actions based on the observations and the model of the environment. In order to minimize the damage incurred by the agents and maximize the information gained, we next provide a solution to the patrolling actions that agents needs to choose.

3 Single-Agent Patrolling

In this section, we first define the information gathering agents and the POMDP framework. Then we present the belief state of reduced dimension. After that, we propose a predictive heuristic and an online single-agent algorithm.

3.1 Patrolling Agents

An patrolling agent is a physical mobile entity capable of gathering information, and maybe damaged by the threat when visiting a vertex. The set of all patrolling agents is denoted as $A = \{1, \ldots, |A|\}$. At each time step $t \in T$, each agent is positioned at a given vertex in graph G. The movement of each agent is atomic, i.e. takes place within the interval between two subsequent time steps, and is constrained by layout graph G, i.e. an agent positioned at a vertex $v \in V$ can only move to a vertex $v' \in adj_G(v)$ that is adjacent to v in graph G. The speed of the agents is assumed to be sufficient to reach an adjacent vertex within a single time step.

3.2 The POMDP Framework

The state at time step t is a tuple $\{v(t), S(t)\} = \{v(t), [s_R^1(t), \ldots, s_R^N(t)], [s_I^1(t), \ldots, s_I^N(t)]\}$, where $v(t)$ is the current position, $s_R^i(t) \in R$ and $s_I^i(t) \in I$ are the threat state and information state at vertex. We assume that $v(t)$ is deterministic and only determined by the destination of the action. While, $S(t)$ follows the discrete-time Markov process with $M = K_1^N K_2^N$ states. Note that $S(t)$ is not directly observable by the agent.

In this POMDP framework, the agent selects an adjacent vertex $v_j \in adj_G(v_i)$ to visit in the next time step as an action, where v_i is the current position. When visiting vertex v_i at time step t, the agent observes $\theta(t) = [s_R^i(t), s_I^i(t)]$, where $s_R^i(t)$ is the

[1] In practice, these transition probabilities may be unknown and the threat state maybe even controlled by an adversary, but it is beyond the scope of this paper.

threat state and $s_I^i(t)$ is the information state at the current vertex. The information state at v_i reset to I_1 immediately. The instantaneous reward at time t is then:

$$\mathcal{R}(t) = \alpha f(s_I^i(t)) - (1-\alpha)h(s_R^i(t)) \tag{3}$$

where $\alpha \in [0,1]$. The objective is to choose the movement action sequentially to maximize the total expected reward accumulated over $|T|$ steps. We now have a POMDP since, the environment state $S(t)$ cannot be fully observed due to the limited sensing ability of agents. Hence, we model the belief vector of the states subsequently.

In our model, the states are not directly observable. The belief vector $B(t) = [b_1(t), \ldots, b_M(t)]$ is defined as the posterior probability distribution for each state, where $b_i(t)$ the conditional probability the environment state is at the ith state at the current time step t.

For any t, the belief vector is a sufficient statistic for the design of the optimal action for next time step [12]. A policy π specifies the action will execute in any given belief state. An optimal policy π^* is a policy by which the agent gets the maximum total expected reward accumulated over $|T|$ steps.

Solving POMDPs is computationally prohibitive except for problems with small state and action space [13], as the dimension of belief vector increases exponentially with the number the vertices. Although the optimal policy may be computed offline and stored before starting the patrol mission, it is still difficult to adapt to changes of the models of environments. To address this, we propose an online method by introducing a reduced belief vector and develop a predictive heuristic to reduce the search space while produce high quality solutions (as we show later).

3.3 Belief State of Reduced Dimension

As the threat state and information state variables at each vertex evolve independently and v is deterministic, we can find a sufficient statistic for the optimal policy whose dimension linearly grows with N, similar to [14, 15]. We introduce the reduced belief state and its transition function in this section.

We define a sufficient statistic belief vector at time t as the conditional probability (conditioned on the observation and decision history) $\Omega(t) = [\Omega_R(t), \Omega_I(t)]$ where:

$$\begin{cases} \Omega_R(t) = (w_R^1(t), \ldots, w_R^N(t)) \\ w_R^n(t) = (w_{R1}^n(t), \ldots, w_{RK_1}^n(t)) \\ \Omega_I(t) \ = (w_I^1(t), \ldots, w_I^N(t)) \\ w_I^n(t) \ = (w_{I1}^n(t), \ldots, w_{IK_2}^n(t)) \end{cases} \tag{4}$$

where $w_{Rk_1}^n(t)$ is the condition probability that the threat state of vertex v_n is R_{k_1}, $w_{Ik_2}^n(t)$ is the condition probability that the information state of vertex v_n is I_{k_2}. Then $\Omega(t)$ is a sufficient statistics of optimal decision making. By exploiting the statistical independence among vertices, we reduce the dimension of the sufficient statistic from $(K_1+K_2)^N$ to $(K_1+K_2)N$. This allows us to reduce the computational and storage complexity of the optimal patrolling policy significantly.

Initially, we assume that the agent is at a specified vertex $v(0)$, and we have initial probabilistic information about the state of each of the N vertices $[\Omega_R(1), \Omega_I(1)]$.

Then, he belief vector $\Omega(t+1)$ and position $v(t)$ for $(t > 0)$ are conditioned on the observations and the action history. The transition $\delta\left(v(t-1), \Omega(t)|a(t), \theta(t)\right)$ from position $v(t-1)$ and belief state $\Omega(t)$ upon an action $a(t) = v_i \in adj\left(v(t-1)\right)$ and observation $\theta(t)$ can be generated as follow:

$$\begin{aligned} &v(t) = v_i \\ &w_R^n(t+1) = \begin{cases} P_{Rk} & \text{if } v_n = v_i, s_R^i(t) = R_k \\ w_R^n(t)P_R & \text{if } v_n \neq v_i \end{cases} \\ &w_I^n(t+1) = \begin{cases} P_{I1} & \text{if } v_n = v_i \\ w_I^n(t)P_I & \text{if } v_n \neq v_i \end{cases} \end{aligned} \tag{5}$$

where $R_k \in R$, $I_k \in I$ and $v_n \in V$.

Based on the transition function above, a policy π specifies a sequence of actions $\pi = [\pi(1), \pi(2), \ldots]$, where $\pi(t)$ is a map from $\Omega(t)$ and $v(t-1)$ to $v(t) \in adj_G(v(t-1))$ for time step t. Given this, the optimal policy can be computed:

$$\pi^* = \arg\max_{\pi} \mathbb{E}^{\pi}\left[\sum_{t=1}^{\infty} \gamma^t \mathcal{R}^{\pi(t)}\left(\Omega(t)|\Omega(1), v(0)\right)\right] \tag{6}$$

where $\mathcal{R}^{\pi(t)}\left(\Omega(t)\right)$ is the reward obtained when the belief state is $\Omega(t)$, $\gamma \in [0, 1]$ is the discount factor.

Although the dimensionality of the belief state is reduced, the problem is still a POMDP and finding the optimal solution is intractable. Based on this reduced belief vector, we develop a predictive heuristic and present the online single-agent algorithm that implements this heuristic in the next section.

3.4 The Predictive Heuristic

Here, we develop a predictive heuristic by: i) introducing the concept of stochastic dominance and add the assumption that the Markov state transition matrixes are monotone matrixes, ii) based on the monotonicity of the transition matrixes, the relationship of the belief states between different vertices become "predictable" without observations, iii) defining the predictive heuristic.

Stochastic Dominance Here, we introduce the concept of stochastic dominance. Stochastic dominance is a central theme in a wide variety of applications in economics, finance and statistics [17]. Stochastic dominance $\succ$ between two K dimension probability vector x, y is defined as $x \succ y$, if:

$$\sum_{j=i}^{K} x(j) \geq \sum_{j=i}^{K} y(j), \quad \text{for } i = 2, 3, \ldots, K \tag{7}$$

Monotonic Assumption We assume that the Markov information model and Markov threat model are monotonic matrixes, i.e. the matrix of transition probabilities P_R and P_I satisfies:

$$\begin{aligned} P_{RK_1} \succ P_{RK_1-1} \succ \ldots \succ \quad P_{R_1} \\ P_{IK_2} \succ P_{IK_2-1} \succ \ldots \succ \quad P_{I_1} \end{aligned} \tag{8}$$

If the matrix of transition probabilities P_R and P_I satisfy the assumption above, then P_R and P_I are monotone matrixes [16]. Under this assumption, if $w_I^n(t) \succ w_I^{n'}(t)$, then $w_I^n(t)P_I \succ w_I^{n'}(t)P_I$. From (5), we know that probability vectors for information states of two vertices keep the relationship of stochastically dominance when no agent visits any of them. Obviously, if $w_I^n(t) \succ w_I^{n'}(t)$, then $w_I^n(t)F \geq w_I^{n'}(t)F$, which means that a stochastically dominant information belief vector is likely to have a higher information value. The same is true that a stochastically dominant threat belief vector is likely to have a higher damage value. In particular, as the information state at a given vertex will reset to I_1 when there is an agent visiting this vertex, the belief vector of information states $(1, 0, \ldots, 0)$ is stochastically dominated by the belief vector of any vertex which is not being visited, so the more recently visited vertex always has a lower expected information value.

Then given on the Monotonic Assumption, the relationship between the belief states at different vertices become "predictable" without observations. We can thus predict the expected reward agents may get from one vertex of the graph when visiting it at a near future step.

Hence, we denote a feasible policy of length D at time t as $\pi_D(t) = (\pi_{t+1}, \ldots, \pi_{t+D})$, which consists of D consecutive deterministic vertices/actions.

Predictive Heuristic Here, we define the predictive heuristic as the predictive expected future reward $\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))]$ for policy $\pi_D(t)$, which is the aggregate of the expected reward of each step in $\pi_D(t)$ as follows:

$$\begin{aligned}\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))] = \sum_{i=1}^{D} \gamma^t \big(&\alpha \hat{w}_I^{\pi_{t+i}}(t+i)F \\ &-(1-\alpha)\hat{w}_R^{\pi_{t+i}}(t+i)L\big)\end{aligned} \tag{9}$$

where, $[\hat{w}_I^{\pi_{t+i}}(t+i), \hat{w}_R^{\pi_{t+i}}(t+i)]$ is the predictive belief vector at the vertex π_{t+i} and time $t+i$. For the step $t+1$, we can get the predictive belief vector $[\hat{w}_I^{\pi_{t+1}}(t+1), \hat{w}_R^{\pi_{t+1}}(t+1)]$ by the current belief vector $\Omega(t)$, current action $a(t)$ and observations $\theta(t)$, i.e. $\Omega(t+1) = \delta\left(\Omega(t)|a_t^*, \theta(t)\right)$, which is the belief vector at $t+1$ and obtained from Equation 5. And for $\{t+2, \ldots, t+D\}$, we get the predicted belief vector based on a transition which omits observations in Equation 5 as follows:

$$\begin{aligned}\hat{w}_R^n(\tau+1) &= \hat{w}_R^n(\tau)P_R \\ \hat{w}_I^n(\tau+1) &= \begin{cases} P_{I1} & \text{if } v_n = \pi_\tau \\ \hat{w}_I^n(\tau)P_I & \text{if } v_n \neq \pi_\tau \end{cases}\end{aligned} \tag{10}$$

where $\tau = \{t+1, \ldots, t+D-1\}$.

Given the predictive heuristic and policies that looks ahead D time periods, the agent compares all feasible paths of length D and chooses the next location to visit according to the path that gives the highest predictive expected reward gained over that path. The details of how to use the heuristic in our online single-agent algorithm is presented in the next section.

Algorithm 1 Single-Agent Patrolling

```
Require: $P_R$: the Markov risk model
Require: $P_I$: the Markov information model
Require: $\Omega(t)$: the belief state of current time step
Require: $\theta^{(t)}$: the observations at the current position
Require: $v(t)$: current position, i.e $a^*(t)$.
Ensure: $a^*(t+1)$: next action of the agent
    ▷ Step $0$: get all feasible policies $\Pi_D(t)$;
    ▷ Step $1$: computing best policy:
 1: for $\pi_D(t) \in \Pi_D(t)$ do
    ▷ Step $1.1$: Get predictive belief state for next $D$ steps:
 2:     $\Omega(t+1) \leftarrow \delta\left(\Omega(t)|v_t, \theta(t)\right)$
 3:     for $\tau \in \{t+1, \ldots, t+D-1\}$ do
 4:         for $v_n \in V$ do
 5:             $\hat{w}^n(\tau+1) \leftarrow \hat{\delta}\left(\hat{w}^n(\tau)|\pi_\tau(\tau)\right)$
 6:         end for
 7:     end for
    ▷ Step $1.2$: Compute the predictive reward for $\pi_D(t)$:
 8:     $\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))] = \alpha w_I^{\pi_{t+i}}(t+i)F + \beta w_R^{\pi_{t+i}}(t+i)L$
    ▷ Step $1.3$: Compare $\pi_D(t)$ with the stored best policy:
 9:     if $\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))] > \mathbb{E}[\hat{\mathcal{R}}(\pi_D^*(t))]$ then
10:         $\pi_D^*(t) \leftarrow \pi_D(t)$
11:     end if
12: end for
    ▷ Step $2$:return the next action from the best policy $\pi_i^*$
13: return $a^*(t+1) \leftarrow \pi_{t+1}^*$
```

3.5 The Online Algorithm

Based on the predictive heuristic, we propose the online algorithm for single-agent patrolling problem (Algorithm 1) in this section.

First, we compute $\Pi_D(t)$, which is the set of all the feasible policies that start from current position $v(t)$ (*step* 0). Then, we compute the predictive expected reward for all the policies. For each policy, the belief state at $t+1$ is updated by the belief state, position and observations at t (line 2) and the predictive belief state at $\{t+2, \ldots, t+D\}$ is computed by Equation 10 (line 3-7). Given this, we compute the predictive reward for the policy (line 8). Thus, the best policy is:

$$\pi_D^*(t) = \arg\max_{\pi_D(t)} \mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))] \tag{11}$$

The best next action here is computed as $a^*(t+1) = \pi_{t+1}^*$, which is the first action of best policy (line 13).

Having defined the online single-agent algorithm for our formulation of patrolling under uncertainty and threats, we extend it to compute policies for multi-agent problems next.

4 Multi-Agent Patrolling

In this paper, we assume all the agents can share their collected observations with each other. Thus, team of agents may not only obtain more information about the environment, but each agent may also make decisions given threat observations are shared by other agents. On the other hand, when more than one agent is positioned at one same vertex, only one information value is obtained for the team but each agent suffers the same damage as if one agent was visiting. Given the two assumptions above, we design an online multi-agent algorithm to coordinate the agents in their patrolling tasks.

First, we note that, while the state variable described in Equation 4 can be used to express the belief state for a multi-agent POMDP, the action space of the POMDP is the Cartesian product of the action spaces of the individual agents. Hence, in so doing, the size of the action space grows exponentially with the number of agents $|A|$, allowing only the smallest of problem instances to be solved. Instead, sequentially computing policies for individual agents as in our multi-agent algorithm avoids this problem of computing a joint policy for the team.

Similar methods have been successfully used to solve multi-agent problems [7, 8]. As these formulations are different from our POMDP, a straightforward application of their methods is not possible. Hence, we consider how to make up the deficiency of sequentially computing policies for individual agents in partially observable problem using our online single-agent algorithm.

When sequentially computing policies for individual agents using our predictive heuristic, there implicitly exists an order in which the agents make actions; agent 1 completes D step actions of its best policy, agent 2 second, etc.. The expected future reward of a policy $\pi^i_D(t)$ of agent i is conditioned on both position $v_i(t)$ belief vector $\Omega(t+1)$ and the best policies of the previously computed policies of agents $A_{-i} = \{1, \ldots, i-1\}$.

The best online patrolling policy for agent i in a multi-agent setting is recursively defined as:

$$\begin{aligned}
&\widehat{\pi}^*_1 = \arg\max\nolimits_{\widehat{\pi}_1} \mathcal{R}'\left(v_1(t), \Omega(t+1)\right)\\
&\widehat{\pi}^*_2 = \arg\max\nolimits_{\widehat{\pi}_2} \mathcal{R}'\left(v_2(t), \Omega(t+1), \widehat{\pi}^*_1\right)\\
&\vdots\\
&\widehat{\pi}^*_i = \arg\max\nolimits_{\widehat{\pi}_i} \mathcal{R}'\left(v_i(t), \Omega(t+1), \widehat{\pi}^*_1, \ldots, \widehat{\pi}^*_{i-1}\right)
\end{aligned} \tag{12}$$

where we use $\widehat{\pi}^*_i$ denotes the best policy of agent i.

To ensure the reward function only takes into account marginal reward value, we need to exclude double counting. There are two types of double counting. First, synchronous double counting, which occurs when two agents patrol the same cluster within the same time step. In this case the reward for patrolling the vertex is received twice. Second, asynchronous double counting, which occurs when agent i design to visit vertex v at t_1, and there was an action of visiting v_n by agent j $(j < i)$ at t_2 $(t_1 < t_2)$ during the D horizon.[4]

[2] Here, the situation is the agent j will visit vertex v_n after agent i. For the situation agent j visits vertex v_n before agent i (i.e. $t_1 \geq t_2$) could be dealt with directly when calculating $\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))]$ in Equation 9.

Without loss of generality, we consider the situation that only v_n in $\pi_D^i(t)$ of agent i has been visited by agent j. If more than one agent of A_{-i} has an action to visit v_n, we assume the time t_2 is nearest[5]to t_1. Based on this assumption, we can see that the expected information reward of agent j for visiting vertex v_n is overestimated, as it is unaware that the i will reset the information at the time t_1. Thus, we introduce a penalty $\hat{p} \in \mathbb{R}^+$ for agent i that compensates for the reduction of reward of agent j, as follows:

$$\mathcal{R}'_i \left(v_i(t), \Omega(t+1), \widehat{\pi}_1^*, \ldots, \widehat{\pi}_{i-1}^*\right) = \mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))] - \hat{p} \tag{13}$$

where $\mathbb{E}[\hat{\mathcal{R}}(\pi_D(t))]$ is the expected reward function defined in Equation 10, and $\hat{p}$ is the loss incurred by agent j that will visit the vertex v_n after i, which is defined as follows:

$$\hat{p} = \hat{r}_{\text{expected}} - \hat{r}_{\text{revised}} \tag{14}$$

where the $\hat{r}_{\text{expected}} \in \mathbb{R}^+$ is the item that agent j compute the expected reward of visiting vertex v_n and the $\hat{r}_{\text{revised}}$ is the revised expected reward of agent j visiting vertex v_n computed by agent i with considering its own action. We define the revised expected belief states at vertex v_n and between time $[t_1+1, \ldots, t_2]$ are $\{\tilde{w}^n(t_1+1), \ldots, \tilde{w}^n(t_2)\}$, which are obtained by the transition Equation 10 based on the predictive belief state $\hat{w}^n(t_1)$ and action $a(t_1) = v_n$. Then the revised expected reward is as follows:

$$\hat{r}_{\text{revised}} = \gamma^{t_2} \left(\alpha \tilde{w}_I^n(t_2) F - (1-\alpha) \tilde{w}_R^n(t_2) L\right) \tag{15}$$

Now using the algorithm to compute the policy of length D as before, we obtain an action for each individual agent. A team action is formed by combining these individual actions. This team action is not optimal, as the policy of agent i is computed greedily with respect to the policies of agents A_{-i}. However, we can still bound the the performance guarantees compared with the policy obtained by searching the joint action space.

We use the following theorem [18] to obtain a bound on the value of the greedily selected policies:

Theorem 1. *Let $f : 2^E \to \mathbb{R}$ be a non-decreasing submodular set function. The greedy algorithm that iteratively selects the element $e \in E$ that has the highest incremental value with respect to the previously chosen elements $I \in E$:*

$$e = \underset{e \in E \setminus I}{\arg\max}\, f(e \cup I) - f(I) \tag{16}$$

until the resulting set I has the desired cardinality k, has an approximation bound $\frac{f(I_G)}{f(I^*)}$ at least $1 - \left(\frac{k-1}{k}\right)^k$, where$I^* \in E$ is the optimal subset of cardinality k that maximises f.

For the number of agents $|A|$ in our formulation, the approximation bound of the greedy algorithm is $1 - \left(\frac{|A|-1}{|A|}\right)^{|A|}$. [7] showed that this approximation bound is monotonically decreasing with $|A|$, and Thus, for $|A| \to \infty$, the multi-agent policy yields at

[3] Only the nearest one needs to be taken into account and this can be deduced from the transition Equation 10.

least $\approx 63\%$ the reward as the best policy obtained by searching the joint policy space for $|A|$ agents.

Having formulated the problem and designed both single-agent and multi-agent algorithms, we will evaluate our methods in the next section.

5 Empirical Evaluation

To empirically evaluate our approach, we applied it to a large graph, which contains 350 vertices and 529 edges. In the paper, the single-agent algorithm could be seen as a special case of the multi-agent algorithm. So we just present the results of the multi-agent algorithm here. In this graph, we simulated two scenarios:

- **Scenario A:** we use same Markov information model and threat model for every vertex;
- **Scenario B:** we attribute 3 different information models and threat models to different vertices in the graph.

We assume $\gamma = 0.9$, $\alpha = 0.33$, $F = [0 \quad 1 \quad 2 \quad 3 \quad 4]$, $L = [0 \quad 1 \quad 2]$. In the Scenario A, we define the two Markov chains as follows:

$$P_R = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.4 & 0.4 & 0.2 \\ 0.0 & 0.2 & 0.8 \end{bmatrix} \tag{17}$$

$$P_I = \begin{bmatrix} 0.8 & 0.1 & 0.1 & 0 & 0 \\ 0.2 & 0.7 & 0.0 & 0.1 & 0 \\ 0.1 & 0.1 & 0.7 & 0.1 & 0 \\ 0 & 0.0 & 0.1 & 0.8 & 0.1 \\ 0 & 0 & 0.1 & 0.1 & 0.9 \end{bmatrix} \tag{18}$$

As we can see, the transition function P_R and P_I satisfies Monotonic Assumption of Equation 8. In Scenario B, we attribute several different Markov models to different vertices.

We measured the total reward of the information value and the damage suffered, and we benchmark our algorithm against Greedy and Random be more precise:

- **Random** moves to a random location adjacent to the agents' current position.
- **Greedy** moves to the adjacent location with the highest value in the next step. When more than one agents are positioned at one same vertex, only one information value is obtained for the team but each agent suffers the same damage as if one agent was visiting. We assume the greedy algorithm sequentially computes greedy policies for individual agents to avoid different agents selecting the same vertex, which is similar to PH-1.
- **PH-D** is our multi-agent patrolling algorithm and we adjust the lookahead parameter D to different values to investigate the value of the extra computation involved for higher values of D. We illustrated the results of our algorithms of different D.

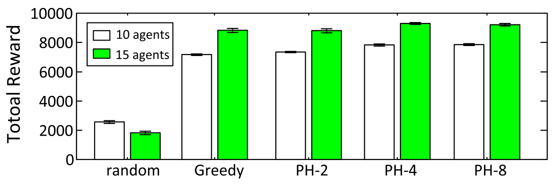

Fig. 1. The result of the scenario with the same model for each vertex.

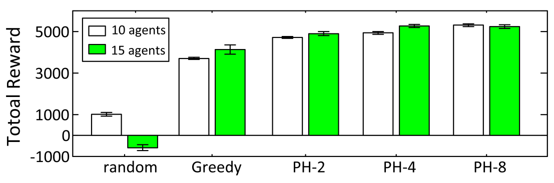

Fig. 2. The result of the scenario with different models for each vertex.

The initial locations of the agents are randomly distributed in the graph. Agents patrol continuously for 3000 time steps in the stochastically changing graph. For each scenario and each algorithm we ran 20 rounds and plotted the results[6] in Figures 1 and 2. In both scenarios, random algorithm performs poorly and its total reward never reaches more than 30% of the reward obtained by the other two algorithms. In Scenario A, both PH-8 and Greedy perform well, and PH-8 outperforms than greedy algorithm by at least 5%. However, for the graph with different Markov models in Scenario B, our algorithm is significantly better than all the other algorithms, and PH-8 outperforms greedy algorithm by more than 43% for 10 agents and by 21% for 15 agents.

6 Conclusion

In this paper, we developed an online multi-agent patrolling algorithm for large partial observable stochastic environment where the information are distributed with threats. A predictive heuristic is defined to evaluate the policies of looking ahead several steps. For the multi-agent algorithm, we extended the sequential policy computation method for individual agents to deal with partially observable problems. We empirically showed that for 10 agents in a large graph, our algorithm outperforms greedy by more than 43%. As this is the first algorithm for patrolling with uncertainty and threats, we intend to study a better heuristic and algorithms that provide theoretical performance guarantees in future work.

[6] The error bars depict the 95% confidence intervals around the means. Non-overlapping error bars invalidate the null hypothesis with $\alpha = 0.05$.

7 Acknowledgements

This work was partially supported by the EPSRC ORCHID project (EP/I011587/1). Mr. Chen would like to thank the China Scholarship Council for sponsoring his visiting study in University of Southampton, Hunan Provincial Innovation Foundation For Postgraduate (No.CX2013B013), and National University of Defense Technology for Outstanding Graduate Innovation Fund (No.B130302).

References

1. Agmon, N., Kraus, S., Kaminka, G. A., and Sadov, V.: Adversarial Uncertainty in Multi-Robot Patrol. In IJCAI. 1811-1817 (2009)
2. Basilico, N., and Gatti, N.: Automated abstractions for patrolling security games. In: AAAI (2011)
3. An, B., Kempe, D., Kiekintveld, C., Shieh, E., Singh, S., Tambe, M., and Vorobeychik, Y.: Security games with limited surveillance. In: AAAI (2012)
4. An, B., Brown, M., Vorobeychik, Y., and Tambe, M.: Security games with surveillance cost and optimal timing of attack execution. In: AAMAS (2013)
5. Vorobeychik, Y., An, B., Tambe, M., and Singh, S.: Computing Solutions in Infinite-Horizon Discounted Adversarial Patrolling Games. In: ICAPS (2014)
6. Lin, K. Y., Atkinson, M. P., Chung, T. H., and Glazebrook, K. D.:A graph patrol problem with random attack times. Morgan Kaufmann. Operations Research (2013)
7. Stranders, R., de Cote, E. M., Rogers, A., and Jennings, N. R.: Near-optimal continuous patrolling with teams of mobile information gathering agents. Artif. Intell. 195,63-105 (2013)
8. Singh, A., Krause, A., and Kaiser, W. J.: Nonmyopic adaptive informative path planning for multiple robots. In: IJCAI. 1843-1850 (2009)
9. Balakirsky, S., Carpin, S., Kleiner, A., Lewis, M., Visser, A., Wang, J., and Ziparo, V. A.: Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue. Journal of Field Robotics. 24, 943-967 (2007)
10. Berger, J., Boukhtouta, A., Benmoussa, A., and Kettani, O.: A new mixed-integer linear programming model for rescue path planning in uncertain adversarial environment. Computers & OR. 39:3420-3430 (2012)
11. Yehoshua, R., Agmon, N., and Kaminka, G. A.: Robotic adversarial coverage: Introduction and preliminary results. In: IROS. 6000-6005 (2013)
12. Smallwood, R. D., and Sondik, E. J.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research. 21,1071-1088 (1973)
13. Papadimitriou, C. H.: Computational complexity. John Wiley and Sons Ltd. (2003)
14. Zhao, Q., Tong, L., Swami, A., and Chen, Y: Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. IEEE Journal on Selected Areas in Communications. 589-600 (2007)
15. Ouyang, Y., and Teneketzis, D.: On the optimality of a myopic policy in multi-state channel probing. Computational complexity. Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on. 342-349 (2012)
16. Keilson, J., and Kester, A.: Monotone matrices and monotone Markov processes. Stochastic Processes and their Applications. 5, 231-241 (1977)
17. Sandholm,W. H.: Orders of limits for stationary distributions, stochastic dominance, and stochastic stability. Theoretical Economics. 5, 1-26 (2010)
18. Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L.: An analysis of approximations for maximizing submodular set functionsI. Mathematical Programming. 14, 265-294 (1978)