Decentralized Patrolling Under Constraints in Dynamic Environments

Shaofei Chen, Feng Wu, Lincheng Shen, *Member, IEEE*, Jing Chen, and Sarvapali D. Ramchurn

***Abstract*—We investigate a decentralized patrolling problem for dynamic environments where information is distributed alongside threats. In this problem, agents obtain information at a location, but may suffer attacks from the threat at that location. In a decentralized fashion, each agent patrols in a designated area of the environment and interacts with a limited number of agents. Therefore, the goal of these agents is to coordinate to gather as much information as possible while limiting the damage incurred. Hence, we model this class of problem as a transition-decoupled partially observable Markov decision process with health constraints. Furthermore, we propose scalable decentralized online algorithms based on Monte Carlo tree search and a factored belief vector. We empirically evaluate our algorithms on decentralized patrolling problems and benchmark them against the state-of-the-art online planning solver. The results show that our approach outperforms the state-of-the-art by more than 56% for six agents patrolling problems and can scale up to 24 agents in reasonable time.**

***Index Terms*—Decentralized patrolling, multiagent systems, planning under uncertainty and constraints, transition-dependent partially observable Markov decision process (TD-POMDP).**

I. Introduction

MULTIPLE unmanned aerial vehicles (UAVs) are increasingly being used to carry out situational awareness tasks in the aftermath of major disasters [1]–[5], such as the Haiti earthquake of 2010 and typhoon Haiyan in 2013. In a disaster response system [6], satellite imagery and crowdsourced reports from members of the public can be provided as low quality *a priori* information about the situation. Given this, emergency response agencies determine a set of specific locations where further information is needed or where should be continuously monitored. UAVs are then deployed to gather recent and high quality information at the designated locations as quickly as possible in order to support an ongoing operation. However, such patrolling problems are often liable to a high degree of dynamism (e.g., fires may spread, wind direction may change) and uncertainty (e.g., it may not be possible to completely observe the causes of fires or locations of casualties may not be exactly known), and may also contain a number of hazards or threats for the UAVs (e.g., UAVs may fly close to buildings on fire or debris may fall on the UAVs). Hence, the key challenges in such patrolling problems are twofold. First, the UAVs cannot cover the entire area at all times, so the dynamics and uncertainty of the monitored phenomena need to be identified to predict the parts of environmental conditions that cannot be sensed directly. Second, the UAVs must coordinate their behaviors in order to collect the most informative measurements as accurately as possible while limiting the damage to the UAVs caused by threats. The work in [7] has considered these challenges and proposed a centralized approach for multiagent patrolling. However, this centralized fashion may not be desirable in disaster response situations because it creates a single point of failure, thereby increasing the vulnerability of the information stream.

In this paper, we focus on decentralized patrolling problems where each agent has its own patrolling area and each area may overlap with some others. Disaster responders typically choose to deploy in such ways to minimize the risk of collisions and to satisfy air traffic regulations. Given this feature, we show that agents only have to interact with small subsets of the other agents to achieve global welfare, demonstrating sparse interactions (i.e., each agent has limited interactions with a small number of neighbors). The coordination of multiple agents with sparse interactions is typically addressed as a distributed constraint optimization problem (DCOP) [8]. In particular, as an approximation approach, the max-sum algorithm [9] has shown to be effective and efficient for solving large DCOPs.

However, on its own, max-sum does not cater for the uncertainty underlying the operation of UAVs. In turn, decentralized partially observable Markov decision process (Dec-POMDP) [10] offers a framework for sequential multiagent decision-making under uncertainty, which also characterizes incomplete or partial information of the environment and other agents due to limited or no communication.

Manuscript received April 13, 2015; revised September 23, 2015 and December 1, 2015; accepted December 3, 2015. The work of S. Chen was supported in part by China Scholarship Council for sponsoring his visiting study in the University of Southampton, in part by the Hunan Provincial Innovation Foundation for Postgraduate under Grant CX2013B013, in part by the National University of Defense Technology for Outstanding Graduate Innovation Fund under Grant B130302, and in part by the National Natural Science Foundation of China under Grant 61403411. The work of L. Shen and J. Chen was supported by the National Natural Science Foundation of China under Grant 61403411. The work of S. D. Ramchurn was supported by the Engineering and Physical Sciences Research Council through ORCHID Programme under Grant EP/I011587/1. This paper was recommended by Associate Editor Q. Shen.

S. Chen, L. Shen, and J. Chen are with the College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China (e-mail: chensf005@163.com; sc16g13@ecs.soton.ac.uk).

F. Wu is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China.

S. D. Ramchurn is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2505737

However, the high complexity makes the scalability of Dec-POMDPs limited [10] (see Section II-B for more details). Moreover, there is relatively little research on constrained Dec-POMDP [11], capturing situations where there is one criterion (reward) to be maximized while keeping other criteria (costs) below the prescribed thresholds during the process.

Against this background, we propose a new model for decentralized patrolling under uncertainty and threats and develop a novel algorithm to solve this model. In more detail, we first represent the patrolling problem with a graph, where the information and the threat at each location (vertex) are independently modeled as multistate Markov chains (which captures the nonstationary[1] feature), whose states are not observed until the location is visited by an agent (which captures the partially observable[2] feature). Then, we cast the planning problem as a constrained Dec-POMDP and solve it with an efficient online algorithm.[3]

Given this, we first propose transition-decoupled POMDP with health constraints (TD-POMDP-HCs), our decentralized formulation of multiagent patrolling under health constraints in nonstationary environments, which is a special case of constrained Dec-POMDP. We then design scalable online algorithms that incorporate Monte Carlo tree search (MCTS) [12] and the max-sum algorithm [13] to exploit sparse agent interactions. In more detail, this paper advances the state-of-the-art in the following ways.

1) We propose the first formal model for decentralized patrolling under uncertainty and threats. Our formulation not only captures the partially observable and nonstationary features of the dynamic environment and the health status of the patrolling agents, but also explicitly accounts for the fashion of decentralized patrolling.
2) We design scalable decentralized online algorithms based on MCTS and max-sum to solve TD-POMDPs. The novelty of this approach lies in the design that each agent constructs a look-ahead tree and these agents expand and update their own trees in a decentralized manner by passing messages to each other.
3) We empirically evaluate our algorithms in simulations and show that they outperform the benchmark by more than 56% for six agents patrolling problems and can scale up to 24 agents in reasonable time.

The rest of this paper is structured as follows. First, we present the background and the related work in Section II and introduce the process of situational awareness in disaster response in Section III. We then define the model of the patrolling problem in Section IV and formulate the decision-making problem as a TD-POMDP-HC in Section V. Next, Section VI describes the decentralized online planning algorithm. Given this, we discuss the variations of our model and algorithm in Section VII and evaluate these approaches in Section VIII. Finally, we conclude this paper in Section IX.

[1]Nonstationary indicates that the joint state distribution of underlying state of the environment does not change over time.

[2]Partially observable indicates that agents cannot directly observe the underlying state of the environment.

[3]In contrast to offline planning algorithms where the whole plan is computed offline, "online planning algorithm" is a standard term in artificial intelligence referring to the case where the agent interleaves planning with execution in a single decision cycle.

II. Background and Related Work

In this section, we first review the literature on multi-UAV patrolling. Then, we present the background on the TD-POMDP model and other similar frameworks. Furthermore, we describe the MCTS approach to solving POMDPs that we reuse later.

A. Multi-UAV Patrolling

A number of coordination algorithms have been developed for UAVs to continuously collect and provide up-to-date situational awareness [3], [14], [15]. Given dynamic environments, previous works [3], [14] consider fully observable (agents can directly observe the underlying state of the environment) stationary models (the joint state distribution does not change over time). A partially observable nonstationary model has been proposed by Ny *et al.* [15], where an agent can only perceive the exact state at its current position. However, these approaches do not consider the health status of agents and the damage that agents may suffer while patrolling.

The work in [7] has considered a general problem of multiagent patrolling under threats, whose method runs in a centralized fashion. It is worth noting that our formulation mainly extends [7] to decentralized patrolling and casts the problem as a TD-POMDP-HC, which is a special case of constrained Dec-POMDP. Compared with the centralized approach in [7], there is no central point of failure and no communication bottleneck in our decentralized patrolling. However, to the best of our knowledge, developing scalable approaches to solve Dec-POMDPs is still an open problem. Fortunately, models of Dec-POMDPs with sparse interactions have been shown to be more scalable than general Dec-POMDPs and our problem has the nature of sparse interaction. We next review the models of Dec-POMDPs with sparse interactions.

B. Transition-Decoupled POMDPs

Dec-POMDP is a natural extension of Markov decision process (MDP) and partially observable MDP (POMDP) to cooperative multiagent settings. While several approaches have been proposed on the scalability of solvers for POMDPs [16]–[18], the computational complexity of solving a Dec-POMDP is significantly higher than that of a POMDP (NEXP-complete [10] versus PSPACE-complete [19]). However, many practical applications suggest that agents have sparse interactions, which show better scalability while preserving optimality. For example, several models have been proposed to capture the sparsity of interactions. Transition-independent reward-dependent models have been developed for transition-independent DEC-MDP [20] and network distributed POMDPs [21], where agents only influence one another through their reward functions. Transition-decoupled POMDPs (TD-POMDPs) model the problems with weakly-coupled transition-dependent agents [22].

TD-POMDPs provide a more natural representation of interactions. However, current approaches for Dec-POMDPs with sparse interactions generate solutions in an offline fashion and no algorithms consider constrained problems. Therefore, we consider the patrolling problems with sparse interactions represented as TD-POMDPs with constraints and consider online methods using a simulator. We next review how MCTS solves general POMDPs in an online fashion.

C. Monte Carlo Tree Search for POMDPs

We design MCTS-based online algorithms in this paper as MCTS has been shown to outperform other planning approaches in many challenging problems [23]. In particular, partially-observable Monte Carlo planning (POMCP) [12], an MCTS-based algorithm, has been successfully applied for single-agent online planning in large POMDPs. POMCP runs by repeatedly performing simulation trials originating from the current belief to incrementally build a look-ahead tree. An unweighted particle filter is used to approximate the belief state. Simulations are performed by using the partially observable upper confidence bounds for trees algorithm [12]. For every history h (history of actions and observations) encountered during a simulation, actions are selected to maximize the upper confidence bounds. Moreover, POMCP has been shown to perform well in large domains with a limited number of simulations and its convergence is satisfied as long as the samples are drawn from the true belief state [12]. In addition, POMCP has been extended by [24] in order to solve multi-agent POMDPs (MPOMDPs), which assume all agents fully communicate with each other to share their view of the world. Instead, our problem only allows the agents to communicate with their adjacent neighbors.

III. Situational Awareness in Disaster Response

This section introduces the process of situational awareness in disaster response. In particular, we explain the process of using *a priori* information to generate a set of targets for UAVs to patrol.

As introduced in [6], emergency response agencies are typically hierarchical, military-style organizations that employ the observe-orientate-decide-act framework [25], [26], which is a well established information gathering and decision making process for deployments in dynamic environments. The main objectives of the response efforts are decided by decision makers at the strategic level, while the allocation of resources and tasks are decided by a command team at the tactical level.

For situational awareness in disaster response, the process of UAV mission planning (as shown in Fig. 1) is designed in collaboration with emergency responders such as Rescue Global and Hampshire County Council. First, prior information about the environment is collected from various resources, and then used to help the commanders to generate targets where more detailed information needs to be collected by UAVs. In particular, maps of roads and key amenities in the disaster area and weather reports can help the commanders to predict the operation situation of the UAVs. Moreover, crowdsourced reports are gathered from online crowdsourcing platforms such as

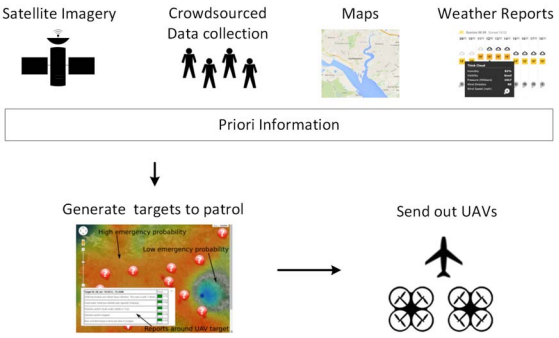

Fig. 1. UAV mission planning for situational awareness in disaster response.

Twitter[4] or Ushahidi.[5] Combined with satellite imagery, this information provides low quality *a priori* information. Given this, the commanders designate a set of distributed targets, then dispatch the UAVs to continuously patrol and gather recent and high quality information at these targets. Hence, this paper features the problem of how to use the limited number of UAVs to execute patrolling in the dynamic and uncertain environment where information is distributed alongside threats.

Thus, we have presented the process of situational awareness in disaster response, and illustrated the function of UAVs patrolling in this process. The following sections define and solve the patrolling problem.

IV. Patrolling Problem

This section introduces our patrolling problem by defining the physical environment and patrolling agents. In particular, we build upon the model of the patrolling environment defined in [7] and extend it by defining the decentralized interaction and cooperative performance of patrolling agents.

A. Patrolling Environment

The physical patrolling environment is defined by its spatial, temporal, and dynamic properties. In particular, in the aftermath of a disaster, a number of specific sites might need urgent attention and access to these sites may be limited to certain areas (e.g., due to trees, debris, or natural obstacles). Hence, we can capture such features in terms of paths along which agents can travel from one disaster site to another. Specifically, the spatial property of the environment is encoded by a graph, which specifies how and where agents can move.

Definition 1 (Graph): We model the environment as an undirected graph $G = (V, E)$. Spatial coordinates of the positions V are embedded in Euclidean space and edges E encode the possible movements between them. We denote the number of the vertices as N, i.e., $N = |V|$.

In disaster response, each disaster site (or target) is a vertex in the graph, and a traversable area between a pair of sites is an edge of the graph. Given the process of situational awareness that was introduced in Section III, the graph can be created by the commanders based on the prior information.

[4]http://www.twitter.com
[5]http://www.ushahidi.com

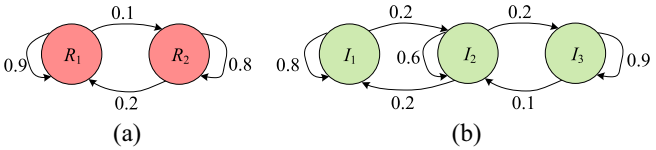

Fig. 2. Example of information and threat models at a vertex. (a) Threat model with two states (i.e., R_1 and R_2). (b) Information model with three states (i.e., I_1, I_2, and I_3), where the probabilities that each information/threat state changes to another over a time step are given (e.g., the probability that R_1 changes to R_2 is 0.1).

Definition 2 (Time): Time is modeled by a set of time steps $\{1, 2, \ldots, T\}$ and at each time step $t \in \{1, 2, \ldots, T\}$ the agents visit some sites in the environment.

To capture the dynamic attributes of the environment, we assume that each vertex holds two states: one for information and one for threats.

Definition 3 (Information State Variable): An information state variable e_v^I indicates different levels of the information at a given vertex $v \in V$.

For example, how many people need help and what is the status of a bridge are information state variables in the disaster response scenario.

Definition 4 (Threat State Variable): A threat state variable e_v^R reflects the level of damage an agent suffers when visiting a given vertex $v \in V$.

For example, the level of fire and the degree of smog are typical threat state variables in disaster response.

Definition 5 (Markov Model of Information and Threat): The two state variables at each vertex change over time according to discrete-time multistate Markov chains.

To capture the transitions of the state variables, we employ a Markov chain model. Specifically, for a Markov chain with K states $S = (S_1, S_2, \ldots, S_K)$, the matrix of transition probabilities for pairs of states is defined as

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1K} \\ p_{21} & \cdots & p_{2K} \\ \vdots & \ddots & \vdots \\ p_{K1} & \cdots & p_{KK} \end{bmatrix}$$

where p_{ij} is the probability that threat state S_i transitions to S_j in one time step and $S_i, S_j \in S$. Next, we present the Markov models of information and threat. An example of the information and threat models at a vertex is shown in Fig. 2. Thus, Fig. 2(a) shows a threat model with two states (i.e., R_1 and R_2) and Fig. 2(b) shows an information model with three states (i.e., I_1, I_2, and I_3), where the probabilities that each information/threat state changes over a time step are given (e.g., the probability that R_1 changes to R_2 is 0.1).

The set of information states $I^n = \{I_1^n, I_2^n, \ldots, I_{K_I^n}^n\}$ for location v_n corresponds to K_I^n stages of information values which agents obtain when visiting a given location. The value of information is determined by the function $f^n : I^n \to \mathbb{R}^+$. The information state at a given vertex independently evolves as a K_I^n-state Markov chain model with a matrix of transition probabilities P_I^n.

Similarly, the set of threat states $R^n = \{R_1^n, R_2^n, \ldots, R_{K_R^n}^n\}$ indicates K_R^n threat levels of vertex $v_n \in V$. The "damage" that an agent suffers when visiting vertex v_n is captured by the function $c^n : R^n \to \mathbb{R}^+$. The threat state at a given vertex independently evolves over time as a K_R^n-state Markov chain and the matrix of transition probabilities is P_R^n.

To bootstrap the information and threat models, we rely on system operators' understanding of the unfolding situation. Specifically, in disaster response, responders typically collect some information about the environment from various resources (such as satellite imagery, weather reports, and crowdsourced reports) before patrolling. Hence, this prior information can be used to build a statistical model of information and threats by various machine learning techniques (such as independent Bayesian classifier combination [27] and Gaussian process) [28]. Although this statistical model is only our initial guess of the true models, it is still helpful to make an initial schedule for the agents. Once more accurate information is obtained and the model is updated accordingly, our online planning method is able to adjust the agents' patrolling schedules based on the new model. This is indeed one of the advantages of our online planning algorithm.

Having modeled the environment in which the agents operate, we next elaborate on the agents' behaviors and goals.

B. Patrolling Agents

Patrolling agents are situated in the patrolling environment defined above.

Definition 6 (Patrolling Agent): A patrolling agent (agent for short) is a physical mobile entity capable of gathering information, and may get damaged by the threat when visiting a vertex. The set of all agents is denoted as $\mathcal{M} = \{A_1, \ldots, A_{|\mathcal{M}|}\}$.

Definition 7 (Patrolling Area): Each agent $A_m \in \mathcal{M}$ has a relatively small designated patrolling area $g_m = (V_m, E_m)$, which is a subgraph of G. Each patrolling area overlaps with some other patrolling areas.[6]

Two examples of the division of the patrolling areas are shown in Fig. 3.

Definition 8 (Health Budget): We define a limited health budget $\beta_m \in \mathbb{R}$ for agent $A_m \in \mathcal{M}$ during the patrolling process lasting T time steps.

The movement and visit capabilities of the agents are defined as follows.

Definition 9 (Movement): When patrolling in a graph G, each agent A_m is positioned at a given vertex in g_m at each time step t. Multiple agents can occupy the same vertex.[7] The movement of each agent is atomic, i.e., takes place within the interval between two subsequent time steps, and is constrained by g_m, i.e., agent A_m positioned at a vertex $v_i \in V_m$ can only move to a vertex $v_i' \in \text{adj}_{g_m}(v_i)$ that is adjacent to v_i in g_m. And we assume that $\forall v_i \in V, v_i \in \text{adj}_{g_m}(v_i)$, i.e., an agent can also

[6]The patrolling areas overlap with each other because different agents may be equipped with different types of sensors and some areas need to be patrolled with different types of sensors to gather different but related datasets.

[7]For example, each vertex can be a room that can host multiple patrolling robots. This is useful if they have different types of sensors or some objects need to be observed simultaneously from different angles.

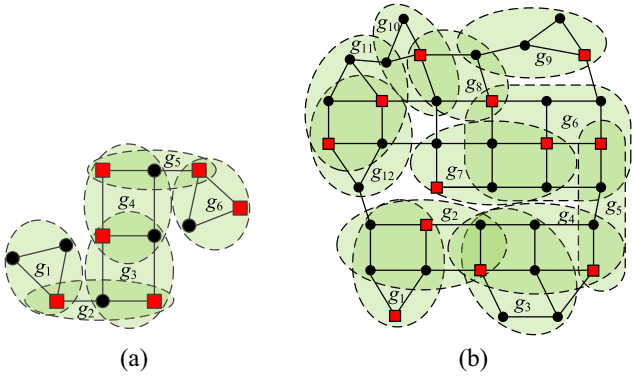

Fig. 3. Example of (a) six agents patrolling and (b) 12 agents patrolling, where each envelope covers the patrolling area g_m of each agent A_m and square vertices are the current positions of agents.

stay at the same vertex. The speed of the agents is sufficient to reach an adjacent vertex within a single time step.[8]

Definition 10 (Visit): Each agent A_m visits its current vertex v_n at each time step. On the one hand, this makes the agent aware of the current information and threat state at v_n, such as I_i^n and R_j^n, respectively. On the other hand, this agent obtains a reward $f^n(I_i^n)$, and suffers a loss $c^n(R_j^n)$ from its health budget β_m. The time it takes to visit a vertex is assumed to be negligible.

As the states at each vertex change over time and agents can only access the exact states at the vertices that they visit, the patrolling environment can be considered nonstationary (i.e., joint probability distribution of its states may change when shifted in time) and partially observable.

The interaction and coordination among the agents can be described as follows.

Definition 11 (Neighbor): A neighbor of agent A_m is an agent whose patrolling area overlaps with that of A_m. We denote $\mathcal{M}_m \subseteq \mathcal{M}$ as the set of neighbors of agent A_m and assume that $A_m \in \mathcal{M}_m$.

Definition 12 (Communication): Each agent A_m can only communicate with its neighbors.

Definition 13 (Cooperative Performance): To value the cooperative performance of gathering information with different numbers of agents at the same time, we assume that there is a function $\alpha : \{0, \ldots, |\mathcal{M}|\} \rightarrow [0, 1]$. Then $\alpha(n)$ denotes the amount of information that can be obtained by n agents for one time step, where these agents are visiting this location together and $n \in \{0, \ldots, |\mathcal{M}|\}$.

Thus, all the agents need to coordinate with each other based on their observations in order to optimize cooperative performance. Specifically, the goal of the agents is then to gather as much information as possible while limiting the damage incurred within the health budget.

Given the definition of patrolling environment and patrolling agents, we summarize the patrolling problem as the following scenario. The group of patrolling agents $\mathcal{M}$ is gathering information over the $N > |\mathcal{M}|$ locations in the patrolling environment. While the agents obtain the information at a location, they may suffer attacks from the threat at that location. The information and threat at each location are independently evolving as multistate Markov chains, and the agents only know the state at the locations with certainty when some agent actually visits it. However, given the transition matrices of these Markov chains, the agents can estimate the states of the locations that are not visited by any agent at a given period. Through this model, the agents continuously select the locations to visit based on the current situation of the agents and the environment.

Having defined the patrolling problem, we now need a decentralized approach for each agent to plan its sequential patrolling actions based on the history of actions and observations of its neighbors and the model of the environment. Hence, in what follows, we propose the formulation TD-POMDP-HC for patrolling within a graph and then design algorithms to solve it.

V. TD-POMDP-HC Model

In this section, we first propose the TD-POMDP framework for multiagent patrolling without constraints. Given this, we then incorporate health constraints into the framework.

A. TD-POMDP Formulation

We now set up our problem of multiagent patrolling in a graph as a TD-POMDP $\langle \mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, r, T \rangle$ as follows.

1) $\mathcal{M} = \{A_1, \ldots, A_{|\mathcal{M}|}\}$ is the set of agents. $\mathcal{M}_m \subseteq \mathcal{M}$ is the set of neighbors of agent $A_m \in \mathcal{M}$.
2) $\mathcal{S}$ is the set of states, which models the positions of the agents and the information and threat states at all the vertices in the environment. There is a particular grouping of state features into local features that make up an agent's local state. We denote A_m's local state as s_m, which is comprised of two sets: $s_m = \langle \bar{e}_m, \bar{v}_m \rangle$. The set of uncontrollable features $\bar{e}_m = \langle (e^R_{m_1}, e^R_{m_2}, \ldots), (e^I_{m_1}, e^I_{m_2}, \ldots) \rangle$ contains the information and threat states at the vertices of the patrolling area g_m for agent A_m. $\bar{e}_m$ is not controllable by any agent, but may be observable by multiple agents.[9] The set of local features $\bar{v}_m$ contains the positions of agents $\mathcal{M}_m$. $\bar{v}_m$ is controlled not only by agent A_m, but also by the other agents in $\mathcal{M}_m$.
3) $\mathcal{A}$ is the set of joint actions. $\mathcal{A} = \times_{1 \leq m \leq |\mathcal{M}|} \mathcal{A}_m$, where $\mathcal{A}_m$ is the set of actions of agent A_m. Each joint action a is defined by $\langle a_1, \ldots, a_{|\mathcal{M}|} \rangle$, where $a_m \in \mathcal{A}_m$. We denote $a_{\mathcal{M}_m}$ as the joint action of agents $\mathcal{M}_m$. Agent A_m selects an adjacent vertex in its patrolling area to visit or "do nothing"[10] as an action a_m.
4) $\mathcal{O}$ is the set of joint observations. $\mathcal{O} = \times_{1 \leq m \leq |\mathcal{M}|} \mathcal{O}_m$, where $\mathcal{O}_m$ is the set of observations of agent A_m. Each joint observation o is defined by $\langle o_1, \ldots, o_{|\mathcal{M}|} \rangle$,

[8] If the distance between two vertices is too far for the agents to reach with its maximal speed, we can simply add another vertex between them where no information needs to be collected.

[9] Examples for uncontrollable features of other problems include time-of-day or temperature.

[10] An agent can only perform the action do nothing when it is dead, which will be discussed later.

where $o_m \in \mathcal{O}_m$. An observation of agent A_m is o_m, which captures the position of A_m and the information and threat state at that position. o_m could be seen as a part of s_m.

5) $\mathcal{T}$ is the set of joint transition probabilities. $\mathcal{T}(s'|s,a) = \prod_{1\le m\le|\mathcal{M}|} \mathcal{T}_m(s'_m|s_m, a_{\mathcal{M}_m})$, where $\mathcal{T}_m(s'_m|s_m, a_{\mathcal{M}_m}) = \mathcal{T}_m(e'_m|e_m)\,\mathcal{T}_m(\bar{v}'_m|\bar{v}_m, a_{\mathcal{M}_m})$ is the local transition function of A_m. Based on the Markov models defined in Section IV-A, we know that $\mathcal{T}_m(e'_m|e_m)$ follows the discrete-time Markov process. $\mathcal{T}_m(\bar{v}'_m|\bar{v}_m, a_{\mathcal{M}_m}) = 1$ if s'_m is the destination of the action $a_{\mathcal{M}_m}$, otherwise $\mathcal{T}_m(\bar{v}'_m|\bar{v}_m, a_{\mathcal{M}_m}) = 0$.
6) Ω is the set of joint observation probabilities. $\Omega(o|s,a) = \prod_{1\le m\le|\mathcal{M}|} \Omega_m(o_m|s_m, a_m)$ is the probability of jointly observing o after taking joint action a in joint state s. As an observation o_m of agent A_m is directly a part of some local states, the observation probability $\Omega_m(o_m|s_m, a_m) = 1$ if o_m is consistent with the corresponding part of s_m and $\Omega_m(o_m|s_m, a_m) = 0$ otherwise.
7) $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the decomposable reward function. $r(s,a) = \sum_m r_m(s_m, a_m)$ is the reward of taking action a in state s. We define $r_m(s_m, a_m) = \alpha(n_{v_i})/n_{v_i} f^i(e^I_i)$ as the information value obtained by agent A_m, where $v_i \in s_m$ is the current position of A_m and n_{v_i} is the number of agents visiting v_i and $\alpha(n_{v_i})$ denotes the amount of information that can be obtained by the agents at this position. Then, we can get the reward of all agents $r(s,a) = \sum_m r_m(s_m, a_m) = \sum_{v_i\in\vec{v}} \alpha(n_{v_i}) f^i(e^I_i)$,[11] which is the sum of the information values obtained by all the agents, where $\vec{v}$ is the set of all the current positions of all the agents.
8) T is the planning horizon.

A history h is represented as a sequence of joint actions and observations of all the agents. A solution to TD-POMDP is specified as a joint policy $\bar{\pi} = \langle \pi_1, \ldots, \pi_{|\mathcal{M}|}\rangle$, where π_m (agent A_m's policy) maps the joint history of agents $\mathcal{M}_m$ to a probability distribution over the actions of agent A_m. The joint policy value function $V^\pi(h)$ is the expected reward accumulated from time t onward when following policy π.

Theorem 1: The value function of our TD-POMDP can be factored as

$$V^\pi(h) = \sum_{m=1}^{|\mathcal{M}|} V_m^{\pi_{\mathcal{M}_m}}\left(h_{\mathcal{M}_m}\right) \tag{1}$$

where $\pi_{\mathcal{M}_m}$ and $h_{\mathcal{M}_m}$ are, respectively, the joint policy and the history of agents $\mathcal{M}_m$, $V_m^{\pi_{\mathcal{M}_m}}(h_{\mathcal{M}_m})$ is the value function of agent A_m, which is the expected reward accumulated from the joint history $h_{\mathcal{M}_m}$ onward when agents $\mathcal{M}_m$ following $\pi_{\mathcal{M}_m}$.

Proof: As the definition of our TD-POMDP model is a special case of the model in [22], in our formulation, agents have local parameters [factored local state $\bar{v}_m$ and rewards $r_m(s_m, a_m)$ for each agent A_m]. However, $\bar{v}_m$ depends on not only the actions of A_m, but also the other agents' actions (movements). As described in [22], the dependency among agents is described using an influence directed acyclic graph (DAG), where each node is an agent. A parent–child relationship in this DAG implies that the child's local states are controllable by the parent, but not vice-versa. Given its parents' policies, an agent can compute its value function. In our representation, this translates into a value factor for each agent A_m, which consists of A_m and all its ancestors in the DAG. The joint-value decomposition along value factors is straightforward. A summary of an agent's influence on its immediate children can be compactly represented as an influence dynamic Bayesian network [22]. Thus, given a specific joint policy, the patrolling environment is redivided into nonoverlapping areas for the agents to patrol. Intuitively, $V_m^{\pi_{\mathcal{M}_m}}(h_{\mathcal{M}_m})$ indicates that the accumulated utility acquired by agent A_m patrolling its area, while the total value $V^\pi(h)$ is the sum of the utilities of all the agents patrolling these nonoverlapping areas. ■

[11] Here, the defined reward only factors the obtained information value. We will formulate the cost of threats in the health constraints of the model in Section V-B.

B. TD-POMDP With Health Constraints

We define a limited health budget β_m for each agent $A_m \in \mathcal{M}$ during the patrolling process lasting T time steps. A history h_m is represented as a sequence of actions and observations of agent A_m. For action a_m and observation o_m, the agent A_m suffers an instantaneous damage $c_m(a_m, o_m) = c^i(e^R_i)$, which is associated with the threat state e^R_i at its position v_i. We denote its remaining health budget for history h_{mt} as

$$b_m(h_{mt}) = \beta_m - \sum_{k=1}^{t} c_m(a_{mk}, o_{mk}). \tag{2}$$

Then, the health constraint of agent A_m needs to be incorporated when making decisions based on the history h_{mt}

$$\text{s.t.} \quad \mathbb{E}^\pi\left[\sum_{k=t+1}^{T} c_m(a_{mk}, o_{mk})\right] < b_m(h_{mt}). \tag{3}$$

Given the above, we propose TD-POMDP-HCs which extends the TD-POMDP formulation in the following ways. First, we augment the local state of agent A_m with a set of variables $\bar{b}_m = \langle b_{m_1}, b_{m_2}, \ldots, \rangle$, where b_{m_i} keeps track of the remaining health budget for agent $A_{m_i} \in \mathcal{M}_m$. b_{m_i} is initialized to the total budget β_{m_i}. Then, in the transition model, when a joint action $a_{\mathcal{M}_m}$ is taken, the damage suffered by agent $A_{m_i} \in \mathcal{M}_m$ will be deducted from the corresponding remaining budget $b_{m_i} \leftarrow (b_{m_i} - c_m(a_{m_i}, o_{m_i}))$. We then augment the action space $\mathcal{A}_m$ of agent A_m with an action do nothing. When $b_{m_i} \le 0$, agent A_{m_i} can only perform the action do nothing and is deleted from the set of agents. The remaining agents continue to patrol until the set of agents is empty or the terminal time point T is reached. We denote the set of remaining budgets of agents $\mathcal{M}_m$ for history $h_{\mathcal{M}_m}$ as $\bar{b}_m(h_{\mathcal{M}_m})$. Thus, the health budget is modeled as a local state of each agent and updated stochastically based on the threat model given the action and observation of the agent. A key goal of a constraint for our patrolling problem is to keep the expected damage below the agent's current health budget (3). However, the expected damage and the future policy of each agent are interdependent (i.e., the expected damage depends

on the agent's future policy and must be subject to the health constraint). Therefore, in our algorithm for TD-POMDP-HC, we use Monte Carlo simulation to generate the policies while testing the health constraints (lines 10–12 of Algorithm 1), which will be introduced in Section VI-B.

Having defined TD-POMDP-HC for multiagent patrolling under health constraints in partially observable nonstationary environment,[12] we next propose a scalable decentralized algorithm to solve TD-POMDP-HC online.

VI. Decentralized Online Planning

In this section, we first define a factored belief representation of the environment state and then propose the transition-decoupled factored belief based Monte Carlo online planning (TD-FMOP) algorithm to solve TD-POMDP-HC.

A. Factored Belief of Environment States

As the local state of each agent is partially observable, the belief state $\Lambda(h_{\mathcal{M}_m}) = \langle \Pr(s_m|_{\mathcal{M}_m})\rangle$ for each agent A_m on joint history $h_{\mathcal{M}_m}$ is a statistic to design an optimal policy of the TD-POMDP-HC. In particular, for local state $s_m = \langle \bar{e}_m, \bar{v}_m, \bar{b}_m\rangle$, the elements of $\bar{v}_m$ and $\bar{b}_m$ are fully observable while the environment elements $\bar{e}_m$ are not directly observable. We can define a direct belief representation of the environment states.[13] In more detail, given history $h_{\mathcal{M}_m}$,[14] the knowledge of the internal state of $\bar{e}$ can be summarized by the belief vector $\Lambda_e(h) = [\lambda_1(h), \ldots, \lambda_{\text{Num\_}e}(h)]$, where $\lambda_i(h)$ is the conditional probability that the environment state $\bar{e}$ is at the ith and $\text{Num\_}e = \prod_{n=1}^{N} K_R^n K_I^n$ is the number of all the environment states. However, the dimension of $\Lambda_e(h)$ grows exponentially with the number N of the vertices, which makes the TD-POMDP-HC computationally prohibitive. Fortunately, a factored belief representation of $\bar{e}_m$ has been proposed in [7] that grows linearly with the number of the vertices and reduces the memory usage dramatically.

Here, we introduce the factored belief of environment states. As the threat and information state variables at each vertex evolve independently, we can find a representation whose dimension grows linearly with N. Specifically, we define a factored belief vector for each represented history $h_{\mathcal{M}_m}$ as the conditional probability $\Psi(h) = [\Psi_R(h), \Psi_I(h)]$, where $\Psi_R(h)$ is defined as

$$\begin{cases} \Psi_R(h) = \left(w_R^1(h), \ldots, w_R^N(h)\right) \\ w_R^n(h) = \left(w_{R1}^n(h), \ldots, w_{RK_R^n}^n(h)\right) \end{cases}$$

Algorithm 1 TD-FMOP

```
1:  procedure SEARCH(ĥ)
2:      for i = 1 → numSamples do
    ▷ Draw a sample of environment state from the factored belief vector.
3:          ē_m ~ Ψ_m(ĥ)
4:          s_m ← (v̄_m(ĥ), b̄(ĥ), ē_m) if
5:          SIMULATE(s_m, ĥ, 0)
6:      end for
    ▷ Coordinate with the other agents to select an action to perform.
7:      return COORDINATE2(T_m(ĥ))
8:  end procedure

9:  procedure SIMULATE(s_m, ĥ, depth)
    ▷ Check that whether the agent dead or not.
10:     if b̄_m <= 0 then
11:         return 0
12:     end if
    ▷ Coordinate with the other agents to select a local joint action.
13:     â* ← COORDINATE1(T_m(ĥ))
    ▷ Use simulator G(s_m, â*) to generate the new state s'_m, the observation ô and the reward r_m.
14:     (s'_m, ô, r_m) ~ G(s_m, â*)
15:     if T_m(ĥâ*ô) == null then
    ▷ Update the remaining health b̄_m by Equation (2).
16:         b̄'_m ← UPDATE(b̄_m)
17:         if b̄'_m <= 0 then
18:             b̄'_m ← 0
19:             DELETE(A_m)
20:         end if
    ▷ Update the belief vector Ψ_m(ĥ) by Equation (4).
21:         Ψ_m(ĥâ*ô) ← UPDATE(Ψ(ĥ), â*, ô)
    ▷ Expand the search tree.
22:         T_m(ĥâ*ô) ← ⟨v̄'_m, b̄'_m, Ψ_m(ĥâ*ô), N_init, V_init⟩
23:         return ROLLOUT(s'_m, ĥâ*ô, depth + 1)
24:     end if
25:     R_m ← r_m + SIMULATE(s'_m, ĥâ*ô, depth + 1)
26:     N_m(ĥ) ← N_m(ĥ) + 1
27:     N_m(ĥâ*) ← N_m(ĥâ*) + 1
28:     V_m(ĥâ*) ← V_m(ĥâ*) + (R_m − V_m(ĥâ*))/N_m(ĥâ*)
29:     return R_m
30: end procedure

31: procedure ROLLOUT(s_m, ĥ, depth)
32:     â* ~ π_rollout(ĥ, ·)
33:     (s'_m, ô, r_m) ~ G(ŝ, â*)
34:     return r + ROLLOUT(s'_m, ĥâ*ô, depth + 1)
35: end procedure
```

where $w_{Rk_1}^n(h)$ is the probability that the threat state at vertex v_n is $R_{k_1}^n$, and $\Psi_R(h)$ is defined as

$$\begin{cases} \Psi_I(h) = \left(w_I^1(h), \ldots, w_I^N(h)\right) \\ w_I^n(h) = \left(w_{I1}^n(h), \ldots, w_{IK_I^n}^n(h)\right) \end{cases}$$

where $w_{Ik_2}^n(h)$ is the probability that the information state at vertex v_n is $I_{k_2}^n$. Then, there are $\sum_{n=1}^{N}(K_R^n + K_I^n)$ elements in each $\Psi(h)$, which grows linearly with N. It has been proved in [7] that $\Psi(h_t)$ is a compact representation of $\Lambda_e(h_t)$ for any h_t to design an optimal solution.

Initially, we have probabilistic information about the state at each of the N vertices. Then, the elements of factored belief vector $\Psi(h)$ are updated to $\Psi(hao)$ upon joint action a and observation o as

$$\begin{aligned} w_R^n(hao) &= \begin{cases} \widetilde{I}_k & \text{if } v_n \in v(hao), s_R^n(h) = R_k^n \\ w_R^n(h)P_R^n & \text{if } v_n \notin v(hao) \end{cases} \\ w_I^n(hao) &= \begin{cases} \widetilde{I}_k & \text{if } v_n \in v(hao), s_I^n(h) = I_k^n \\ w_I^n(h)P_I^n & \text{if } v_n \notin v(hao) \end{cases} \end{aligned} \tag{4}$$

where $\forall v_n \in V$, $R_k^n \in R^n$, $I_k^n \in I^n$, and $\widetilde{I}_k$ is a unit vector with the kth item is 1. P_R^n and P_I^n are, respectively, the matrices of transition probabilities of the threat and information at position v_n. As shown in (4), the threat belief vector $w_R^n(hao)$ at one

[12] To note, we are aiming to solve the patrolling problems instead of a special case of TD-POMDPs because we have relaxed the general assumption of TD-POMDPs (i.e., allowing local communication) and included an external component (i.e., having health budgets), both of which are well motivated from the patrolling problems.

[13] We only define the belief of $\bar{e}_m$ in s_m as it has been shown in [29] that only the not directly observable part of the state has to be represented as a belief to design the optimal policy.

[14] Without loss of generality, we assume that there is $\mathcal{M}_m = \mathcal{M}$ and then $h_{\mathcal{M}_m} = h$, which simplifies the notations in the rest of Section VI-A.

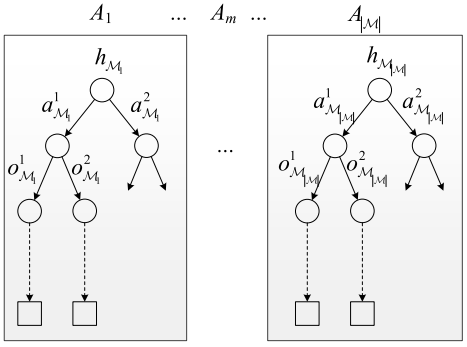

Fig. 4. For TD-FMOP, lookahead trees are constructed in parallel and one for each agent, where each agent A_m keeps track of the joint history, actions, and observations of the group of agents $\mathcal{M}_m$. The agents expand their trees using MCTS while coordinating with the others for action selections.

vertex v_n that some agent visits is updated to $\widetilde{I}_k$ based on the observation $R^n_k(h)$ at this vertex, while $\mathbf{w}^n_R(hao)$ at some other vertex that no agent visit is updated by the current threat belief vectors $\mathbf{w}^n_R(h)$ and the threat Markov model P^n_R at this vertex. For $\forall v_n \in V$, the update of the information belief $w^n_I(hao)$ is similar to $\mathbf{w}^n_R(h)$.

Thus, we have defined the factored belief, which is a compact belief representation for TD-POMDP-HC. Given this, we next propose the decentralized online planning algorithm.

B. TD-FMOP Algorithm

Using the representation of the factored belief vector of the environment and the factored value function (1), we design the TD-FMOP (see Algorithm 1) based on MCTS for multiagent online planning. The bottleneck of directly applying MTCS for multiagent problems is the fact that the number of joint actions and observations are exponential in the number of agents, which leads to a look-ahead tree with a very high branching factor. To combat this problem, we exploit the factored nature of the value function and construct a number of trees in parallel, one for each agent. In more detail, in this section, we first propose the running procedures of TD-FMOP for each agent. Then we detail the stages in which the agents have to coordinate with each other, while introducing the method that the agents using max-sum to pass messages for decentralized coordination. Given this, we discuss the convergence of TD-FMOP.

1) TD-FMOP: Instead of constructing a single tree during MCTS in POMCP for single-agent POMDPs [12], as shown in Fig. 4, we construct $|\mathcal{M}|$ trees in parallel by exploiting the sparse interactions between the agents, which addresses the problem of the large number of joint actions and observations. This technique exploits the structure in multiagent problems and makes MTCS scale to larger Dec-POMDPs in terms of the number of the agents. In this section, we propose the procedures of each agent running TD-FMOP while leaving the coordination stages to be explained in Section VI-B2.

Each node in the search tree of agent A_m is denoted as $T_m(\hat{h}) = \langle \bar{v}_m(\hat{h}), \bar{b}_m(\hat{h}), \Psi_m(\hat{h}), N_m(\hat{h}), V_m(\hat{h})\rangle$,[15] where $\bar{v}_m(\hat{h})$, $\bar{b}_m(\hat{h})$, and $\Psi_m(\hat{h})$ are the sets of positions, the remaining health budgets, and the factored belief vector of agents $\mathcal{M}_m$, $N_m(\hat{h})$ is the count of the node visits, and $V_m(\hat{h})$ is the value. Each node $T_m(\hat{h})$ also contains a value $V_m(\hat{h}\hat{a})$ and a visitation count $N_m(\hat{h}\hat{a})$ for each joint action $\hat{a}$, and the overall count $N_m(\hat{h}) = \sum_{\hat{a}} N_m(\hat{h}\hat{a})$.

The SEARCH procedure (lines 1–8) is called from the current joint history $\hat{h}_t$. Since $\bar{v}_m(\hat{h}_t)$ and $\bar{b}_m(\hat{h}_t)$ are directly observable, the agent only draws a sample of the environment state $\bar{e}_m$ from the factored belief vector $\Psi_m(\hat{h}_t)$ for Monte Carlo simulations (line 3). For every joint history $\hat{h}$ encountered during SIMULATION (lines 9–30), the algorithm checks the remaining health (lines 10–12), selects a joint action $\hat{a}^*$ by COORDINATION1 (line 13),[16] uses the simulator $\mathcal{G}(s_m, \hat{a}^*)$ to generate a new state s'_m, an observation $\hat{o}$ and a reward r_m (line 14), and updates the factored belief vector $\Psi_m(\hat{h})$ by (4) (line 21). When SEARCH is complete, the agent selects the action a^*_m from the joint action $\hat{a}^*$ by COORDINATION2 (line 7) to perform, and receives an observation o_m from the environment.

To deal with the health constraints for TD-POMDP-HC, we use this Monte Carlo simulation to generate policies while ensuring the expected accumulated damage of each agent satisfies the health constraint of (3). In particular, each agent A_m keeps track of the remaining health budget by (2). At each time step of a patrolling operation, each agent checks its health budget and informs its neighbors to delete itself from their interaction lists if its health budget has run out. Similarly, during each Monte Carlo simulation in SEARCH procedure in Algorithm 1, each agent keeps track of the remaining health budget by (2) (line 16). Once its health budget runs out (line 17), an agent should inform its neighbors to remove its actions from their joint actions when expanding the current node in their trees (line 19).

2) Decentralized Coordination: All the agents run TD-FMOP in parallel. Two stages are needed for decentralized coordination: 1) COORDINATE1 (line 13) is selecting actions to search/expand the trees within SEARCH and 2) COORDINATE2 (line 7) is selecting actions to perform after SEARCH.

For each joint history h during a SIMULATE, we denote $\bar{T}(h) = \langle T_1(h_{\mathcal{M}_1}), \ldots, T_{|\mathcal{M}|}(h_{\mathcal{M}_{|\mathcal{M}|}})\rangle$ as the set of visiting nodes of all the $|\mathcal{M}|$ trees. For each search/expand step in each simulation, the agents are searching/expanding their trees in parallel and actions are selected to maximize the upper confidence bounds as

$$a^* = \arg\max_a \sum_{m=1}^{|\mathcal{M}|} U_m(h_{\mathcal{M}_m} a_{\mathcal{M}_m}) \tag{5}$$

where $U_m(h_{\mathcal{M}_m} a_{\mathcal{M}_m}) = V_m(h_{\mathcal{M}_m} a_{\mathcal{M}_m}) + c\sqrt{\log N(h_{\mathcal{M}_m})/N(h_{\mathcal{M}_m} a_{\mathcal{M}_m})}$ and $V_m(h_{\mathcal{M}_m} a_{\mathcal{M}_m})$ is the value of agent A_m that associated with $h_{\mathcal{M}_m}$ and $a_{\mathcal{M}_m}$. After SEARCH, actions are selected by maximizing the

[15] We denote $\hat{h}$ as $h_{\mathcal{M}_m}$, $\hat{a}$ as $a_{\mathcal{M}_m}$, and $\hat{o}$ as $o_{\mathcal{M}_m}$ in Section VI-B1 for short.

[16] As the maximization of the local value does not guarantee a good overall performance of all agents like single-agent MCTS, a coordination approach is needed for action selections of each agent, which will be discussed later.

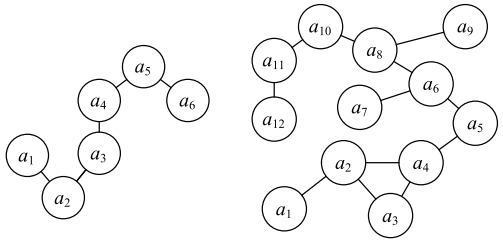

Fig. 5. Corresponding constraint graphs for the scenarios of (a) six agents patrolling and (b) 12 agents patrolling illustrated in Fig. 3, where a connection of two agents' actions denotes that their actions are constrained with each other as they share an overlapping area and need to coordinate to select actions.

joint action value

$$a^* = \arg\max_{a} \sum_{m=1}^{|\mathcal{M}|} V_m\big(h_{\mathcal{M}_m} a_{\mathcal{M}_m}\big). \tag{6}$$

We next show that both (5) and (6) can be cast as DCOPs, and how we can use max-sum to solve them.

3) Action Selection: Here, we define the problem of action selection as a standard DCOP $\langle \mathcal{M}, \mathcal{X}, \mathcal{D}, \mathcal{V} \rangle$ (see [8] for more details of DCOPs), where $\mathcal{M}$ is the set of agents and $\mathcal{X} = \{a_1, \ldots, a_{|\mathcal{M}|}\}$ is the set of action variables, each variable a_m is controlled by exactly one agent. $\mathcal{D} = \{\mathcal{A}_1, \ldots, \mathcal{A}_{|\mathcal{M}|}\}$ is the set of variable domains, and each variable a_m selects its actions from $\mathcal{A}_m$. Then, $\mathcal{V} = \{V_1, \ldots, V_{|\mathcal{M}|}\}$ is the set of functions and we define each function $V_m = V_m(h_{\mathcal{M}_m} a_{\mathcal{M}_m})$, which is the mth item of (5) and depends on $a_{\mathcal{M}_m} \subseteq \{a_1, \ldots, a_{|\mathcal{M}|}\}$. The goal of this DCOP is to find an action variable assignment that maximizes the sum of the functions. Now we have mapped this action selection problem as a DCOP. For scenarios of six agents patrolling and 12 agents patrolling illustrated in Fig. 3, the agents must coordinate their actions to maximize the gathered information in the overlapping areas. Therefore, in the formulated DCOP above, constraints connect the agents that share an overlapping area, and Fig. 5 shows the corresponding interaction graphs.

Algorithms for DCOPs have been successfully applied to numerous multiagent problems [8]. As an approximation algorithm, max-sum [9] has shown to be effective for large DCOPs, and is not exponential in the number of agents. We use max-sum for action selections by (5) and (6) of TD-FMOP. Each agent A_m is responsible for deciding the action selected by its own variable a_m and for passing messages to its neighbors. Thus, although max-sum is approximating the solution to a global optimization problem [i.e., (6)], it involves only local communication and computation. We refer the reader to Appendix to see the full details of DCOPs and how max-sum is implemented in our context.

Note that, using max-sum in Algorithm 1, each agent only has to communicate with its neighbors and the computation of its local value is only conditioned on the histories and policies of its neighbors. Given this, Algorithm 1 can operate in a decentralized manner with limited local information and rapidly converge to an approximately optimal solution, which is much more efficient and effective than full communication algorithms in [24]. In more detail, the agents need to share all of their observations for using the algorithm in [24], and in practice, these observations may contain a large amount of the sensing data. The computation of agents' policy can only be started after receiving all of these observations, which may result in severe delay when using low bandwidth network and the algorithm in [24] then cannot be applied to online patrolling problems. However, the agents only send messages that contain limited local information using our max-sum-based algorithm. Moreover, max-sum is an anytime algorithm,[17] which makes it more suitable for time critical applications that require good enough solutions rapidly as opposed to optimal ones that take a long time to compute. Moreover, in this decentralized approach, there is no central point of failure and no communication bottleneck for our algorithm, and the patrolling solution scales well as the number of agents increases (as we show later).

4) Convergence: Given the structure of TD-FMOP, its convergence is determined by two aspects: 1) MTCS and 2) max-sum. First, the convergence of MCTS for online planning in partially observable situations has been established in [12] that as long as the samples are drawn from the true belief state. This result can be directly extended to our model. Then, max-sum uses a factor-graph representation of the DCOP and this can generate cycles which undermine its convergence properties [30]. max-sum is known to be optimal on acyclic factor graphs but provides no general guarantees on optimality when cycles exist. However, extensive empirical evidence demonstrates that this family of algorithms for DCOPs generates good approximate solutions [9], [30], [31]. To sum up, the convergence of TD-FMOP depends on the performance of max-sum and we show our algorithm can achieve good approximation in the section of experiments.

VII. Variants of TD-FMOP

Having formulated the decentralized model TD-POMDP-HC and its algorithm TD-FMOP, in this section, we propose some variations of this model and this algorithm that we use as benchmarks to compare the performance of our algorithms.

First, we propose TD-POMCP for solving a TD-POMDP-HC by extending POMCP [12] with decentralized coordination of action selections. The difference between TD-POMCP and TD-FMOP is their two ways of representing the belief state: 1) TD-POMCP uses a particle filter [12] and 2) TD-FMOP uses the factored representation. Similar to POMCP, TD-POMCP is a general algorithm for TD-POMDPs and can be applied to problems that are too large or too complex to represent its belief state.

Second, we can view the POMDP-HC as a centralized version of TD-POMDP-HC by assuming full communication across all the agents. As an MPOMDP with complete communication can be reduced to a POMDP with a single centralized controller that takes joint actions and receives

[17] An anytime algorithm is an algorithm that can return a valid solution to a problem even if it is interrupted at any time before it ends.

TABLE I
MODELS AND ALGORITHMS

	Model	Algorithms
Centralised	POMDP-HC	FMOP, FMOP*, POMCP
Decentralised	TD-POMDP-HC	TD-FMOP, TD-POMCP

joint observations [32], a POMDP-HC can be seen as an instance of constrained POMDP. Then, POMCP can be used to solve a POMDP-HC directly by considering its constraints. Furthermore, we can simplify TD-FMOP by centrally selecting an joint action of all the agents and we name this algorithm as FMOP.

Third, we propose FMOP*, an extension of FMOP which balances health considerations in the longer term against information gathering in the shorter term. FMOP defines the parameter τ which determines the number of time steps MCTS looks ahead, while the available horizon of the remaining health budget is $T-t$, where t is the current time. We define a heuristic budget within τ horizon $\tilde{b}_m(h_{mt}) = b_m(h_{mt})(\tau(t)/(T-t))$ for each agent when using FMOP*.

Finally, we illustrate the relations of these models and algorithms in Table I.[18] We empirically evaluate TD-FMOP, TD-POMCP and FMOP and benchmark them with POMCP in the next section.

VIII. EMPIRICAL EVALUATION

We consider a disaster response scenario in which an earthquake happened in a suburban area (see [6] for more details). Given the process of situational awareness that was introduced in Section III, the commanders designate a number of locations for UAVs to patrol. The UAVs then continuously fly among these locations to gather and update their specific information, which is used to assist the commanders for an allocation of the relief tasks (such as digging people out of rubble, moving water treatment units to populated areas, or extinguishing fires). To test the performance of our patrolling algorithms, we use the graphs shown in Fig. 3 that mimic typical patrolling problems in the real-world. Given this, we compare the centralized algorithm FMOP with POMCP and the decentralized algorithm TD-FMOP with TD-POMCP, respectively, where POMCP is the state-of-the-art online POMDP algorithm. For the patrolling scenarios, we define several different three-state information Markov models and two-state threat Markov models for each vertex, and attribute these models, different information and damage value functions to different vertices in the graph of each scenario. We run 100 rounds for each scenario and each algorithm, and evaluate the performance of each algorithm by the average total reward and runtime. The algorithms run on a machine with 2.6 GHz Intel dual core CPU and 1 GB RAM.

A. Evaluation of Factored Belief and Heuristic Budget

To empirically evaluate the performance of the factored belief, we applied POMCP, FMOP and FMOP* to a POMDP-HC of two agents patrolling on a graph of square grid with 12 vertices (3×4), while each agent patrols among all the vertices. FMOP can be seen as combining POMCP with the factored belief representation, while FMOP* extends FMOP with the heuristic budget. The health budgets of the two agents are 100 and 150, respectively. For this two-agent patrolling problem, there are about 5^2 joint actions, 6^2 observations, and approximately 6^{12} environment states. Agents continuously patrol for 200 time steps in the stochastically changing graph and look ahead ten time steps at each time.

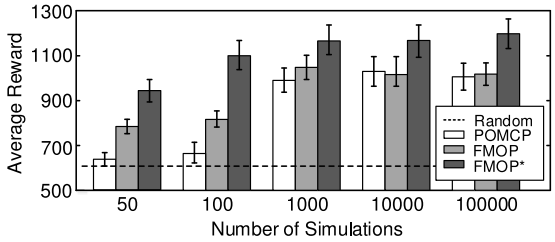

Fig. 6. Average rewards of the problem of two agents patrolling.

TABLE II
RUNTIME(S) IN SIMULATIONS WITH TWO AGENTS

	50	100	1000	10000	100000
FMOP*	0.11	**0.14**	1.18	9.22	74.98
FMOP	0.11	0.18	**1.73**	16.27	223.5
POMCP	0.05	0.09	**3.46**	20.56	217.7

The average total reward of different algorithms with different number of simulations are shown in Fig. 6 and their runtimes are presented in Table II. We first compare FMOP with POMCP. We can see that FMOP outperforms POMCP for different numbers of simulations (by 21.19% with 50 simulations and by 19.03% with 100 simulations), showing that there is some benefit to its exact belief representation. Both FMOP and POMCP are MCTS-based algorithms and their performance grows with the number of simulations. We then compare FMOP* with FMOP and POMCP. Using its health budget management heuristics in FOMP* for this 200 time step horizon problem, the agents can cover more of the environment. The results show that the performance of FMOP* with 100 simulations supersedes the performance of FMOP and POMCP regardless of the number of simulations.

B. Evaluation of Scalability

For TD-POMDP-HC, we empirically evaluate the efficiency and scalability of TD-FMOP, TD-POMCP and FMOP by benchmarking them with POMCP.[19] We construct two scenarios.

1) *Scenario A:* As shown in Fig. 3(a), there are six agents patrolling on a graph with 12 vertices and 15 edges, in which six specific patrolling areas are designated and each agent has about two neighbors. For FMOP and POMCP, the number of joint actions and observations of the six agents are about 3^6 and 6^6, while for TD-FMOP

[18]Note that, although we propose algorithms by introducing how to solve problems with health constraints, it is obvious that these algorithms can also be used to solve problems without constraints.

[19]We assume that agents have full communication when solving TD-POMDP-HC by FMOP or POMCP.

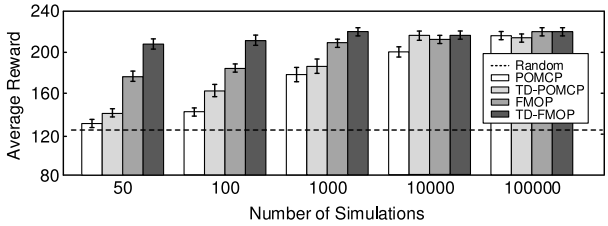

Fig. 7. Average rewards of the problem of six agents patrolling.

TABLE III
RUNTIME(S) IN SIMULATIONS WITH SIX AGENTS

	50	100	1000	10000	100000
TD-FMOP	0.41	**0.88**	9.02	93.26	1059.44
FMOP	0.15	0.27	**3.39**	45.12	468.60
TD-POMCP	0.13	0.39	6.87	**93.38**	1146.23
POMCP	0.09	0.17	3.34	64.04	**588.46**

and TD-POMCP, the number of local joint actions and observations are about 6×3^3 and 6×6^3.

2) *Scenario B:* As shown in Fig. 3(b), there are 12 agents patrolling on a graph with 38 vertices and 59 edges, in which 12 specific patrolling areas are designated and each agent has about three neighbors. For FMOP and POMCP, the number of joint actions and observations are about 3^{12} and 6^{12}, while for TD-FMOP and TD-POMCP, the number of local joint actions and observations are about 12×3^4 and 12×6^4.
3) *Scenario C:* There are 24 agents patrolling on a graph with 76 vertices and 119 edges, in which 24 specific patrolling areas are designated and each agent has about three neighbors. For FMOP and POMCP, the number of joint actions and observations are about 3^{24} and 6^{24}, while for TD-FMOP and TD-POMCP, the number of local joint actions and observations are about 24×3^4 and 24×6^4.

For all scenarios, agents patrol for a ten time step horizon in each graph. For scenario A, we mainly compare the performance of these four algorithms for solving this six-agent patrolling problem, and plot the average reward of different algorithms with different numbers of simulations in Fig. 7, also present their runtimes in Table III. For a specific number of simulations, the performance of distributed approaches TD-FMOP and TD-POMCP obviously exceeds FMOP and POMCP, respectively, while TD-FMOP performs best. For small numbers of simulations, TD-FMOP outperforms TD-POMCP by 46.48% in 50 simulations and by 30.24% with 100 simulations, and achieves more than 90% performance of all these algorithms in 100 000 simulations. TD-FMOP outperforms POMCP by more than 56.72% with less or equal 100 simulations.

We evaluate the scalability of TD-FMOP using scenarios B and C. For these scenarios, the average reward and runtime of each algorithm with different numbers of simulations are shown in Tables IV and V. For scenario C with 24 agents, the average reward of randomly selecting actions is 520.4. As FMOP and POMCP are centralized algorithms, the number of joint actions and observations is too large for them to get a result within 30 min. In contrast,

TABLE IV
RESULTS OF SIMULATIONS WITH 12 AGENTS

	50		100		1000	
	value	time(s)	value	time(s)	value	time(s)
TD-FMOP	**408.2**	**18.06**	412.8	39.78	438.1	578.6
FMOP		–		–		–
TD-POMCP	254.9	3.31	257.5	6.12	**263.2**	**72.51**
POMCP		–		–		–

Time limit violations (30 minutes) are indicated by '–'.

TABLE V
RESULTS OF SIMULATIONS WITH 24 AGENTS

	50		100		1000	
	value	time(s)	value	time(s)	value	time(s)
TD-FMOP	**807.8**	**40.15**	826.7	78.67	843.3	1371.7
FMOP		–		–		–
TD-POMCP	522.5	7.94	524.3	16.08	519.8	172.9
POMCP		–		–		–

Time limit violations (30 minutes) are indicated by '–'.

the runtimes of TD-FMOP and TD-POMCP are reasonable and much lower than those of FMOP and POMCP. This is because of their distributed approximation calculation (i.e., max-sum) for action selection. However, the particle filter of TD-POMCP is not effective at approximating the true belief state any more as the states change stochastically and frequently in a state space which is too large. Given this, the average reward generated by TD-POMCP is close to the result of randomly selecting actions. TD-FMOP outperforms TD-POMCP by more than 58.65% because of its exact belief update.

Thus, we have empirically evaluated the effective of the factored belief and the heuristic budget by comparing the algorithms POMCP, FMOP, and FMOP*, and have shown that our TD-FMOP is scalable to large problems (i.e., 24 agents patrolling).

IX. CONCLUSION

In this paper, we proposed new models for decentralized patrolling under uncertainty and health constraints in partially observable nonstationary environments, the computational complexity of which is too high for state-of-the-art algorithms to solve. We then designed the online planning algorithms with distributed solutions that scale up to a large number of agents. In particular, our algorithms combine the health budget, the factored belief representation, MCTS and max-sum for online planning. Then we empirically showed that our algorithms are efficient in settings with small numbers of agents and can scale up to many agents in settings with sparse interactions.

In real-world disaster response, it is important that UAVs can continuously monitor the environment. As shown, in [5], it is now possible to engineer interfaces that will allow emergency responders, with little knowledge of the algorithmic techniques we employ, to specify tasks and problem spaces for the deployment of multiple heterogeneous UAVs in dynamic and uncertain environments. Hence, we have developed our model to cater for the requirements of the experts who manufacture and deploy UAVs in dangerous environments and who may be able to use such interfaces underpinned by solutions

like ours. This paper is therefore one important foundational step in helping to solve some of the key challenges they face when sending expensive (human) assets to dangerous environments.

Finally, note that our formulation is a very general model for planning under uncertainty such that the solution can be applied to other domains such as exploring physical structure of crash site environment. Although we assume that the complete graph of the sites in our model is known, we do not make any assumptions about the states of the sites. However, it may be possible to use the physical features (or obstacles) in the environment as one of the state variables of each site to model them more accurately and plan for paths around them.

APPENDIX
DISTRIBUTED CONSTRAINT OPTIMIZATION AND MAX-SUM ALGORITHM

Here, we introduce the formulation of DCOPs and how max-sum can solve DCOPs in a decentralized fashion.

A standard DCOP formalization of a multiagent coordination problem is a tuple $\langle \mathcal{M}, \mathcal{X}, \mathcal{D}, \mathcal{V} \rangle$, where $\mathcal{M} = \{A_1, \ldots, A_{|\mathcal{M}|}\}$ is the set of agents and $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{M}|}\}$ is the set of variables, each variable x_m is controlled by exactly one agent A_m. The agent A_m is responsible for assigning values to the variables it owns. $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{M}|}\}$ is a set of discrete and finite variable domains, and each variable x_i can take values in the domain D_i. Then, $\mathcal{V} = \{V_1, \ldots, V_n\}$ is a set of functions that describe the constraints among the variables. Each function $V_j{:}D_{j_1} \times \cdots \times D_{j_{k_j}} \rightarrow \mathbb{R} \cup \{-\infty\}$ depends on a set of variables $\mathbf{x_j} \in \mathcal{X}$, where $k_i = |\mathbf{x_j}|$ is the arity of the function and $-\infty$ is used to represent hard constraints. Each function assigns a real value to each possible assignment of the variables it depends on. The goal is then to find a variable assignment that maximizes the sum of constraints

$$\arg\max_{\mathbf{x}} \sum_{i} V_i(\mathbf{x_i}). \tag{7}$$

DCOPs are usually graphically represented using an interaction graph, where variables are represented as circles and an edge between two variables indicates that the two variables participate in a constraint.

Algorithms for DCOPs have been successfully applied to numerous multiagent problems [8]. As an approximation algorithm, max-sum [9] has shown to be effective for large DCOPs and we then introduce the details of max-sum.

max-sum is a specific instance of a general message passing algorithm that exploits the GDL in order to decompose a complex calculation by factorizing it (i.e., representing it as the sum or product of a number of simpler factors). To use max-sum, a DCOP needs to be encoded as a special bipartite graph, called a factor graph. A factor graph comprises two types of nodes: 1) variable nodes (usually depicted as circles) and 2) function nodes (usually depicted as squares) [14]. Undirected links connect each function to its variables. A factor graph explicitly represents the relationships among the variables and the functions nodes. Considering the situation depicted in Fig. 8 and taking agent A_2 as an example, we

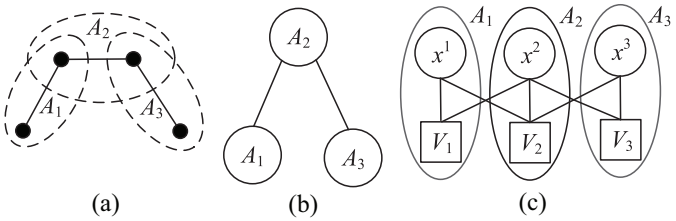

Fig. 8. (a) Each patrolling area of three agents in the environment whose patrolling areas overlap, where black circles are vertices in the graph. (b) Agent interaction graph. (c) Utility-based factor graph representations for the problem.

connect the function node representing V_2 with variable x_2 (the action of agent A_2) and with variables x_1 and x_2 (neighbors' actions). The overall function represented by this factor graph is given by $V = V_1(x_1, x_2) + V_2(x_1, x_2, x_3) + V_3(x_2, x_3)$, which is the social welfare function for the system. As we can see that it can create loops in this factor graph. max-sum is known to be optimal on acyclic factor graphs but provides no general guarantees on optimality when cycles exist. However, extensive empirical evidence demonstrates that this family of algorithms generate good approximate solutions [9], [30], [31].

Notice that by using this formalization there is a clear allocation of variables and function nodes to the agents. In other words, each agent is responsible for deciding the allocation of its own variable, for receiving messages for its function and variable nodes and for updating the messages that flow out of its function and variable nodes. In this way, the agents can negotiate over the best possible actions continuously, thus being able to quickly react to possible changes in the environment. Finally, we refer the reader to [9] for more details of using max-sum.

REFERENCES

[1] I. Maza, F. Caballero, J. Capitán, J. R. Martínez-de-Dios, and A. Ollero, "Experimental results in multi-UAV coordination for disaster management and civil security applications," *J. Intell. Robot. Syst.*, vol. 61, nos. 1–4, pp. 563–585, 2011.

[2] F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings, "Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection," in *Proc. ICRA*, St. Paul, MN, USA, 2012, pp. 469–476.

[3] R. Stranders, E. M. de Cote, A. Rogers, and N. R. Jennings, "Near-optimal continuous patrolling with teams of mobile information gathering agents," *Artif. Intell.*, vol. 195, pp. 63–105, Feb. 2013.

[4] J. Capitan, L. Merino, and A. Ollero, "Decentralized cooperation of multiple UAS for multi-target surveillance under uncertainties," in *Proc. Int. Conf. Unmanned Aircraft Syst.*, Orlando, FL, USA, 2014, pp. 1196–1202.

[5] S. D. Ramchurn *et al.*, "A study of human-agent collaboration for multi-UAV task allocation in dynamic environments," in *Proc. IJCAI*, Buenos Aires, Argentina, 2015, pp. 1184–1192.

[6] S. D. Ramchurn *et al.*, "HAC-ER: A disaster response system based on human-agent collectives," in *Proc. AAMAS*, Istanbul, Turkey, 2015, pp. 533–541.

[7] S. Chen, F. Wu, L. Shen, J. Chen, and S. D. Ramchurn, "Multi-agent patrolling under uncertainty and threats," *PLoS ONE*, vol. 10, no. 6, 2015, Art. ID e0130154.

[8] M. Yokoo, *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems*. Berlin, Germany: Springer, 2012.

[9] A. Farinelli, A. Rogers, and N. R. Jennings, "Agent-based decentralised coordination for sensor networks using the max-sum algorithm," *Auton. Agents Multi-Agent Syst.*, vol. 28, no. 3, pp. 337–380, 2014.

[10] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[11] F. Wu, N. R. Jennings, and X. Chen, "Sample-based policy iteration for constrained DEC-POMDPs," in *Proc. ECAI*, Montpellier, France, 2012, pp. 858–863.
[12] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. NIPS*, Vancouver, BC, Canada, 2010, pp. 2164–2172.
[13] R. Stranders, F. M. Delle Fave, A. Rogers, and N. R. Jennings, "A decentralised coordination algorithm for mobile sensors," in *Proc. AAAI*, Atlanta, GA, USA, 2010, pp. 874–880.
[14] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *Proc. AAMAS*, Estoril, Portugal, 2008, pp. 639–646.
[15] J. L. Ny, M. Dahleh, and E. Feron, "Multi-UAV dynamic routing with partial observations using restless bandit allocation indices," in *Proc. Amer. Control Conf.*, Seattle, WA, USA, 2008, pp. 4220–4225.
[16] G. Shani, R. I. Brafman, and S. E. Shimony, "Prioritizing point-based POMDP solvers," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1592–1605, Dec. 2008.
[17] X. Li, W. K. Cheung, and J. Liu, "Improving POMDP tractability via belief compression and clustering," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 1, pp. 125–136, Feb. 2010.
[18] G. Shani, "Task-based decomposition of factored POMDPs," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 208–216, Feb. 2014.
[19] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Math. Oper. Res.*, vol. 12, no. 3, pp. 441–450, 1987.
[20] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, "Solving transition independent decentralized Markov decision processes," *J. Artif. Intell. Res.*, vol. 22, no. 1, pp. 423–455, 2004.
[21] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs," in *Proc. AAAI*, Pittsburgh, PA, USA, 2005, pp. 133–139.
[22] S. J. Witwicki and E. H. Durfee, "Influence-based policy abstraction for weakly-coupled DEC-POMDPs," in *Proc. ICAPS*, 2010, Toronto, ON, Canada, pp. 185–192.
[23] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[24] C. Amato and F. A. Oliehoek, "Scalable planning and learning for multiagent POMDPs," in *Proc. AAAI*, Austin, TX, USA, 2015, pp. 1995–2002.
[25] T. Grant, "Unifying planning and control using an OODA-based architecture," in *Proc. Annu. Res. Conf. South Afr. Inst. Comput. Sci. Inf. Technol. IT Res. Develop. Countries*, Mpumalanga, South Africa, 2005, pp. 159–170.
[26] T. Grant and B. Kooter, "Comparing OODA & other models as operational view c2 architecture topic: C4isr/c2 architecture," in *Proc. ICCRTS*, Virginia Beach, VA, USA, 2005.
[27] E. Simpson, S. Roberts, I. Psorakis, and A. Smith, "Dynamic Bayesian combination of multiple imperfect classifiers," in *Decision Making and Imperfection*, Berlin, Germany: Springer, 2013, pp. 1–35.
[28] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[29] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1053–1068, 2010.
[30] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[31] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, vol. 7. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[32] D. V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *J. Artif. Intell. Res.*, vol. 16, pp. 389–423, Jun. 2002.

Shaofei Chen received the B.S. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2009, and the M.S. degree in automation from the National University of Defense Technology, Changsha, China, in 2011, where he is currently pursuing the Ph.D. degree in automation.

He was a visiting Ph.D. student with the Electronics and Computer Science Department, University of Southampton, Southampton, U.K., supported by the China Scholarship Council from 2013 to 2015. His current research interests include artificial intelligence, multiagent systems, and automated planning.

Feng Wu received the B.E. and Ph.D. degrees in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2006 and 2011, respectively.

He was a Research Fellow with the University of Southampton, Southampton, U.K., from 2011 to 2014, researching on the ORCHID project. He is currently an Associate Researcher of Computer Science with the USTC. His current research interests include artificial intelligence, multiagent systems, automated planning, and robotics.

Lincheng Shen (M'10) received the B.E., M.S., and Ph.D. degrees in automatic control from the National University of Defense Technology (NUDT), Changsha, China, in 1986, 1989, and 1994, respectively.

In 1989, he joined the Department of Automatic Control, NUDT, where he is currently a Full Professor and serves as the Dean of the College of Mechatronics and Automation. He has published over 100 technical papers in refereed international journals and academic conferences proceedings. His current research interests include mission planning, autonomous and cooperative control, biomimetic robotics, and intelligent control.

Prof. Shen has initiated and organized several workshops and symposia, including the International Workshop on Bionic Engineering 2012 and the Chinese Automation Congress 2013. He has been serving as an Editorial Board Member of the *Journal of Bionic Engineering* since 2007.

Jing Chen received the B.E. and Ph.D. degrees in control science from the National University of Defense Technology (NUDT), Changsha, China, in 1993 and 1999, respectively.

In 1999, he joined the College of Mechatronics and Automation, NUDT, where he is currently a Full Professor and serves as the Head of the Institute of Automation. He has co-authored two books and published over 50 papers in refereed international journals and academic conferences proceedings. His current research interests include mission planning, artificial intelligence, robotics, and autonomic computing.

Sarvapali D. Ramchurn received the B.S. degree in information systems engineering from Imperial College London, London, U.K., in 2001, and the Ph.D. degree in multiagent systems from the University of Southampton, Southampton, U.K., in 2005.

He is currently an Associate Professor of Electronics and Computer Science with the University of Southampton, where he carries out research into the design of autonomous agents and multiagents for real-world socio-technical applications including energy systems, disaster management, and crowdsourcing. He works closely with industry. His current research interests include number of fields, such as machine learning, data science, and game theory. His papers are referred with over 2000 Google Scholar citations and his work has featured in various media, including BBC News, New Scientist, Engineering and Physical Sciences Research Council Pioneer, and Wired.

Dr. Ramchurn was a recipient of multiple best paper awards for his work at top AI conferences.