Centralized Periodic Planning under Asynchronous Communication for Multi-Agent Monitoring

David Fornos[1], Federico Rossi[2], Dylan A. Shell[3], and Daniel Selva[1]

***Abstract*— This paper examines the problem of coordinating the observations of multiple agents constrained to periodic trajectories that communicate asynchronously with a central planner. We are motivated by settings such as active monitoring missions tracking stochastic and spatially spreading events like wildfires or flooding, where a rapid response is essential and the spatial extent can be large. In such cases, "always-on" networking may be infeasible and continuous coordination may be prohibitively costly. Periodic trajectories are a natural constraint for relevant classes of systems, e.g., UAV swarms that cycle around recharging stations or Earth observation satellite constellations; moreover, these lead to recurring communication opportunities with compute-capable infrastructure. We introduce the Multi-Agent Asynchronous Periodic Partially Observable MDP (MA-APPOMDP), a new planning framework that formalizes asynchronous check-in times and centralized but delayed information flow. We propose two algorithms tailored to this new model: the Asynchronous Belief Branching Algorithm (ABBA), which performs exact belief branching over unknown observations, and SB-ABBA, a sampling-based approximation where scalability is prioritized over exactness. Empirical results on different wildfire event monitoring problems show that our methods consistently achieve higher event coverage and lower detection delay than several heuristic and planning baselines, with SB-ABBA scaling to larger problem instances.**

I. INTRODUCTION

Coordinating multiple agents under partial observability is a long-standing challenge in AI planning and robotics [1], [2]. While substantial progress has been made in both centralized and decentralized settings, most frameworks assume either flexible agent mobility or continuous communication. In practice, many real-world systems impose strict *structural constraints* on both motion and communication that make these assumptions unrealistic.

We consider a class of problems where agents are bound to fixed periodic trajectories and only communicate with a central planner during intermittent contact windows, which differ for each agent. Between these opportunities, execution is entirely decentralized: each agent continues along its route following a precomputed script of actions until the next synchronization. Here, the central planner must provide individual plans based on partially outdated beliefs about the global state. The regime described here, illustrated in Figure 1, arises naturally in domains such as Earth-observation (EO) constellations, or UAV swarms for surveillance and environmental monitoring, where continuous coordination is often infeasible due to bandwidth, latency, or power limitations [3], [4]. For Earth observation satellites, motion is constrained by fixed orbital paths that determine their contact opportunities with ground stations for observation or communication. For the UAVs, operations may be constrained to a fixed patrol loop that repeatedly flies over areas of interest, typically implemented as a chain of waypoints that the vehicle must follow [5]. In both cases, the planning problem is to decide where and when to direct each sensor footprint in order to maximize the value of the observations [6], [7]. Imagine a disaster monitoring application where the goal is to detect as many active wildfires as possible in the shortest amount of time. Here, the agents must coordinate to minimize observation redundancy, while observing the most critical areas. In this example, the evolution of the environment is highly structured with spatiotemporal autocorrelation. Thus, the information gathered by one agent will affect the decisions of others, even when they observe distinct regions at different time steps.

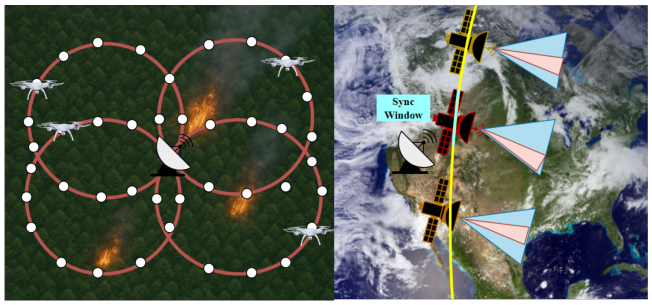

Fig. 1: **Left:** UAVs follow fixed periodic patrol loops and communicate with a ground station periodically when they are in range. **Right:** Earth-observing satellites follow orbital tracks and only communicate with a ground station during contact opportunities. In both cases, coordination must be planned centrally but executed locally under partial and delayed information.

Wildfire monitoring is but one application, but it exemplifies the broader class of planning problems which form the focus of our work. The key characteristic of problems in this class is that multiple agents make decisions in the same environment, but can only communicate with a central planner at known, periodic, asynchronous times. This induces a belief update dynamic where, when the planner chooses a course of action for an agent, it knows what actions other agents have pledged to take, up to some horizon; it does *not* know the outcome of these actions; and it knows when outcomes will be revealed and new partial plans will have to be formed. The periodic nature of the belief update and planning process is the defining feature of the problem.

[1]D. Fornos and D. Selva are with the Department of Aerospace Engineering, Texas A&M University, College Station, TX, 77843; {dfornos,dselva}@tamu.edu.

[2]F. Rossi is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109. federico.rossi@jpl.nasa.gov.

[3]D. Shell is with the Department of Computer Science & Engineering, Texas A&M University, College Station, TX, 77843; dshell@tamu.edu.

In this paper, we introduce the Multi-Agent Asynchronous Periodic Partially Observable MDP (MA-APPOMDP). This framework extends MPOMDPs, Dec-POMDPs, and LOMDPs by explicitly modeling deterministic periodic motion together with asynchronous check-ins. Unlike previous patrolling or swarm-planning formulations, MA-APPOMDP directly accounts for heterogeneous synchronization schedules and delayed propagation of information across agents.

Building on this formalization, we develop two tailored planning algorithms. Firstly, the *Asynchronous Belief Branching Algorithm (ABBA)* is an exact open-loop planner that systematically branches over possible observation outcomes until the next synchronization. While ABBA is generally impractical for problems of realistic size, it offers key insight into the structure of solutions to MA-APPOMDPs. We use this insight to develop SB-ABBA. *SB-ABBA* is a sampling-based approximation inspired by point-based value iteration [8], which achieves tractable scalability.

The remainder of the paper is organized as follows. Section II reviews prior work on MPOMDPs, Dec-POMDPs, LOMDPs, and related formulations for information gathering and active sensing. Section III formalizes the problem, describing the spatio-temporal event dynamics and introducing the MA-APPOMDP framework. Section IV presents our proposed algorithms, ABBA and SB-ABBA. Section V reports empirical results in synthetic environments with comparisons with heuristic baselines. Finally, Section VI concludes with a discussion of limitations and directions for future research.

II. Related Work

A. MPOMDPs, Dec-POMDPs and LOMDPs

The literature on decision making under partial observability is extensive. For a single-agent problem, Partially Observable Markov decision Processes (POMDPs) are the classic model and many methods have been developed to solve them [8], [9], [10], [1]. When the problem is extended to a multi-agent setting, complexity grows substantially, and it is useful to distinguish between coordination mechanisms of various types on the basis of their underlying communication assumptions. At one extreme are Multi-Agent POMDPs (MPOMDPs), which assume the presence of a centralized planner that has access to all agents' observations and can compute a joint policy. This setting is applicable when communication is sufficiently frequent to maintain a common belief over the global state. Typical solution approaches reduce the problem to a single-agent POMDP over the joint action–observation space, so classical methods such as exact dynamic programming [9] or approximate solvers like Point-Based Value Iteration (PBVI) [8], [10] can be applied. Nevertheless, the twin curses of dimensionality and history make exact MPOMDP solvers impractical for all but small problems. At the opposite extreme, if the agents cannot communicate continually and must plan decisions in a decentralized manner, the standard model is the Decentralized POMDP (Dec-POMDP) [2], [11]. Dec-POMDPs are significantly more complex than their centralized counterparts: optimal solutions are NEXP-complete [2], and practical algorithms typically rely on structural assumptions or approximations. Well-known families of solvers include policy search methods such as Memory-Bounded Dynamic Programming (MBDP) [12] and its variants, policy graph search, and point-based extensions to the decentralized setting (e.g., Perseus, PBVI adaptations). More recently, Monte Carlo Tree Search and heuristic search approaches have also been proposed to address larger Dec-POMDPs [11].

There are also many variations on these core models. One particular inspiration for this paper was the LOMDP model [13], an MDP where part of the state can be fully known if the agent is in the appropriate pose as defined by a locality function (e.g., line of sight). Note that most classical POMDP formulations assume rewards that depend on the state and action. However, in many active sensing or target localization tasks, it is preferable to define the reward function over the belief state (e.g., expected information gain [14]).

B. Multi-Agent Information Gathering Problem

Multi-agent information-gathering describes the engagement of multiple robots or vehicles to gather information about an environment that may not be observed directly or completely at all times. Many approaches model information gathering as a decision problem over a grid where each region evolves as an independent Markov chain. Agents only sense the regions they observe, so planning must decide where to observe to reduce uncertainty. Because exact solutions scale poorly, most methods turn to online planning. Zhou et al. [15] describe this as a factored MPOMDP and introduce an online solver that combines POMCP with sequential allocation. Lauri et al. [16] propose an information-theoretic Dec-POMDP in which robots share information during periodic communication windows. Their results show that even limited coordination can sustain cooperative sensing. Zhou et al. [17] study UAV swarms in a continuous patrolling setting. They model event detection as probabilistic and design a two-layer control scheme that links local sensing with centralized planning. This line of work also appears in concrete surveillance tasks. Casbeer et al. [3] analyze the use of small UAVs for forest fire monitoring. Their system assigns patrol routes to aircraft and restricts data upload to times when a vehicle passes near a ground station. Scherer and Rinner [4] extend the idea to persistent surveillance. They focus on how to route UAVs so that information collected during patrols reaches decision makers quickly while still maintaining coverage.

III. Problem Formulation

A. MA-APPOMDPs

Definition 1 (MA-APPOMDP): We begin by defining a Multi-Agent Asynchronous Periodic Partially Observable Markov Decision Process (MA-APPOMDP) as the tuple

$$\mathcal{M}_{\text{MA-APPOMDP}} = \langle \mathcal{N}, S^{\text{pub}}, S^{\text{priv}}, \theta, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \{\mathcal{L}_i\}, P, Z, R, \gamma, b_0, \{H_i\} \rangle$$

where:

- $\mathcal{N} = \{1, \ldots, N\}$ is the set of agents.

- S^{pub} is the joint clock state, represented by the vector $\tau = (\tau_1, \ldots, \tau_N)$. Each $\tau_i \in \{0, \ldots, T_i - 1\}$ denotes the phase of agent i along its periodic trajectory of length T_i, with $\tau_i = 0$ defined as the phase in which agent i is in range of the central planner and can synchronize.
- S^{priv} is the environment state, represented as an event map $x \in \{0, \ldots, M-1\}^K$ over K regions, which evolve according to stochastic dynamics.
- $\theta = (\theta_1, \ldots, \theta_N)$ is the vector of initial phases, which specifies the phase of each agent at $t = 0$.
- $\{\mathcal{A}_i\}$ is the set of actions available to agent i, consisting of observing up to L regions from its footprint $g_i(\tau_i)$ (i.e., the agent's field of regard). See Definition 2 below.
- $\{\mathcal{O}_i\}$ is the set of observations for each agent i, corresponding to the event states of the regions observed from its footprint.
- $\{\mathcal{L}_i\}$ defines the locality function, where $\mathcal{L}_i(k) = \{\tau_i \mid k \in g_i(\tau_i)\}$ is the set of phases in which region k lies inside the footprint of agent i.
- P is the state transition kernel, which evolves the event map according to the underlying stochastic dynamics.
- Z is the observation model, which returns a perfect observation, (i.e., the true state of a region) if observed.
- R is the reward function.
- $\gamma \in [0, 1]$ is the discount factor.
- b_0 is the initial belief over S^{priv}.
- $\{H_i\}$ are the planning horizons for each agent.

Definition 2: The footprint mapping g is defined as a periodic function $g_i : \{0, \ldots, T_i - 1\} \to 2^{\{1,\ldots,K\}}$, which assigns to each clock phase of agent i the subset of regions that lie within its sensing footprint at that phase, also known as the field of regard. Since the trajectory is periodic, g_i repeats with period T_i. Furthermore, we require every region of interest to be observable by at least one agent at some phase of its trajectory: $\bigcup_{i=1}^{N} \bigcup_{\tau_i=0}^{T_i-1} g_i(\tau_i) = \{1, \ldots, K\}$.

As the state is only partially observable, the planner maintains a belief state b_t at each time step t, which consists of the deterministic clock belief b_t^c over public state S^{pub} and the probabilistic environment belief b_t^e over private state S^{priv}. The pair (b_t^c, b_t^e) is the sufficient statistic for planning in the MA-APPOMDP. We denote the belief transition as a general mapping $b_{t+1} = f(b_t, a_t, o_{t+1})$, where a_t is the joint action taken at time t and o_{t+1} is the joint observation received at time $t+1$.

B. Why Open-Loop Plans?

Solving MA-APPOMDP requires full policies that map every possible observation history to actions. This causes a complexity blow-up similar to Dec-POMDPs, which are NEXP-complete [2]. For horizon H, action set $\mathcal{A}$, observation set $\mathcal{O}$, and N agents, the joint policy space has size $|\mathcal{A}|^{N \cdot \frac{|\mathcal{O}|^H - 1}{|\mathcal{O}| - 1}}$, so policy search quickly becomes intractable.

Open-loop plans shrink the search space. An open-loop plan is a fixed sequence of actions of length H, with joint size $|\mathcal{A}|^{NH}$. This is still exponential in H but far smaller than the policy space, which makes planning feasible for larger problems. Open-loop plans also match the problem structure. Agents follow periodic trajectories and revisit regions in a fixed order, so sensing opportunities repeat. A plan that assigns subregions to phases aligns with this periodicity.

Additionally, we assume that agents have limited onboard computational capability and do not perform online planning between synchronization events. This matches real operational architectures such as those discussed in Section I. For example, in an EO satellite constellation, onboard processors are typically used for state estimation, guidance, and low-level autonomy, whereas computationally intensive planning is performed on the ground and uploaded to the satellite during contact windows, often as time-tagged command sequences [18]. Hence, here we restrict attention to open-loop plans and leave reactive policies for future work.

C. Optimization Goal

The planner assigns individual plans to agents at their sync times. Each plan maximizes the expected cumulative reward defined next in Section III-F, where the expectation is taken over the possible observation histories of all agents that remain hidden between their own sync steps. For an agent synchronizing with sync time denoted as t_i^{sync}, we define:

a) Clean time: Let $\sigma = \{t_j^{\text{sync}}\}_{j=1}^{N}$ denote the sync map, the set of last synchronization times for all agents. Each t_j^{sync} is the most recent time when agent j communicated with the planner. The clean time t_{clean} is the latest discrete timestep at which the planner has incorporated information from all agents, i.e., $t_{\text{clean}} = \min_{j \in \mathcal{N}} \sigma_j$. We define $b^e_{t_{\text{clean}}}$ as the joint belief at that time, which is fully determined.

b) Optimization: At t_i^{sync}, the planner assigns agent i an open-loop plan $\pi_i = (a_{i,0}, \ldots, a_{i,H_i-1})$ and optimizes

$$\pi_i^* = \arg\max_{\pi_i} \; \mathbb{E}\left[\sum_{l=0}^{H_i-1} \gamma^l \, R\big(b^e_{t_i^{\text{sync}}+l}, a_{i,l}\big)\right], \tag{1}$$

with $\mathbb{E}$ over $(b^e_{t_i^{\text{sync}}:t_i^{\text{sync}}+H_i-1}) \sim \mathbb{P}\big(\cdot \,\big|\, b^e_{t_{\text{clean}}}, \pi_i, \pi_{-i}, o_{t_{\text{clean}}:t_i^{\text{sync}}}\big)$, where:

- $(b^e_{t_i^{\text{sync}}:t_i^{\text{sync}}+H_i-1})$ are the belief trajectories induced by the candidate plan π_i together with the previously assigned plans of the other agents π_{-i}.
- $\mathbb{P}(\cdot \mid b^e_{t_{\text{clean}}}, \pi_i, \pi_{-i}, o_{t_{\text{clean}}:t_i^{\text{sync}}})$ is the distribution over belief trajectories induced by the initial belief, the belief transition kernel (see sec. III-A), the observation model Z, the agents' plans, and the known observations between t_{clean} and t_i^{sync}.
- $R(b^e_{t_i^{\text{sync}}+l}, a_{i,l})$ is the reward obtained when the system belief is $b^e_{t_i^{\text{sync}}+l}$ and agent i executes action $a_{i,l}$.

D. Environment Representation

To apply this formulation to the multi-agent monitoring problem, we must define what spatio-temporal events are and how their dynamics are modeled. By "spatio-temporal event," we refer to a stochastic phenomenon over space and time, such as fires, oil spills, or disease epidemics. A region affected by an event has a discrete variable representing its status over time (e.g. persisting, spreading to neighbors, growing in intensity, or fading out). Let the environment be represented by a grid with K cells, each with finite state space

$\mathcal{X} = \{0, \ldots, M-1\}$. The full environment state at time t is the random vector $\mathbf{E}(t) = \big(E_1(t), \ldots, E_K(t)\big) \in \mathcal{X}^K$. The exact first-order Markov model on the joint state can be expressed as $P\big(\mathbf{E}(t+1) = \mathbf{e}' \,\big|\, \mathbf{E}(t) = \mathbf{e}\big) = T(\mathbf{e}' \mid \mathbf{e})$, where T is an arbitrary transition kernel on $\mathcal{X}^K$. This simple Markov model can accurately capture the dynamics of many events. However, representing and using it in this form scales as $|\mathcal{X}|^K \times |\mathcal{X}|^K$, which is intractable for large K.

To address this intractability, we assume that events evolve through short-range contagion, which means that a cell's future state depends only on its own current value and those of its immediate neighbors. Specifically, let $G = (V, E)$ be the grid graph of the environment, with $\mathcal{N}(k)$ the neighbors of cell k. The transition kernel factorizes as

$$T(\mathbf{e}' \mid \mathbf{e}) \;=\; \prod_{k=1}^{K} \phi_k\big(e'_k;\; e_k,\; e_{\mathcal{N}(k)}\big) \tag{2}$$

The kernel $\phi_k\big(e'_k;\; e_k,\; e_{\mathcal{N}(k)}\big)$ is assumed to perfectly define the probabilities of a cell k evolving to a specific next state, conditioned on its previous state and the previous state of its neighbors. This defines a Dynamic Bayesian Network (DBN) [19], in which each variable $E_k(t+1)$ has directed edges from its own previous state $E_k(t)$ and from the states of its neighbors $\{E_j(t) : j \in \mathcal{N}(k)\}$. This assumption is commonly used to model spatio-temporal events [20], [21]. The model parameters can be learned from data or informed by physics.

1) Belief Evolution on the Environment: As mentioned above, evolving the environment using a joint state transition kernel is intractable for moderately large grids. Similarly, keeping an exact joint belief over the environment is also infeasible. Thus, we factor the belief over the grid cells, meaning that we keep a distribution for each cell over the state space $\mathcal{X}$. Now, to evolve the factored beliefs while maintaining the locality assumption, we propose the following update rule. Let $b^e_{t,k}(x)$ denote the belief that $E_k(t) = x$. Let U_{t+1} be the set of cells observed at time $t+1$, and let o_k denote the observation obtained for cell $k \in U_{t+1}$. The belief update is given by:

$$b^e_{t+1,k}(x') = \begin{cases} \delta[x' = o_k], & \text{if } k \in U_{t+1}, \\ \displaystyle\sum_{x_k \in \mathcal{X}} \sum_{x_{\mathcal{N}(k)} \in \mathcal{X}^{|\mathcal{N}(k)|}} \phi_k\big(x';\, x_k, x_{\mathcal{N}(k)}\big) \times \\ \qquad b^e_{t,k}(x_k) \prod_{\ell \in \mathcal{N}(k)} b^e_{t,\ell}(x_\ell), & \text{otherwise,} \end{cases} \tag{3}$$

where $\delta[\cdot]$ is the Kronecker delta.

E. Performance metrics

To evaluate the effectiveness of the planners, we measure four metrics. The *Event Observation Percentage* (EOP) quantifies what percentage of events occurring in the grid are successfully detected at least once during the run of the scenario. A cell contains an event if its state is different from NO EVENT. The *normalized detection delay* (NDD) measures how quickly events are observed relative to their expected lifetime. The *final uncertainty* indicates the residual entropy over the grid at the end of the simulation. Finally, we report the *average planning time* required by each algorithm.

F. Reward

Each planner optimizes an additive reward function designed to balance two goals: reducing belief uncertainty and detecting valuable events. Formally, the reward at time t for action a_t given a belief b^e_t is defined as:

$$R(b^e_t, a_t) = \sum_{k \in U_t} \left(w_h\, G(b^e_{t,k}) + \sum_{x \in \mathcal{X}} w_v b^e_{t,k}(x) f(x) \right) \tag{4}$$

where U_t is the set of cells observed at time t by executing action a_t, $b^e_{t,k}$ is the belief of cell k, and $\mathcal{X}$ is the event state space. The first term is the information gain, with $G(b^e_{t,k}) = \mathcal{H}_{\text{prior}}(b^e_{t,k}) - \mathcal{H}_{\text{post}}(b^e_{t,k})$, where $\mathcal{H}(b^e_{t,k})$ is the Shannon entropy of the cell belief before and after the observation. Note that $\mathcal{H}_{\text{post}}(b^e_{t,k})$ will always be zero given the perfect observation model within locality assumption. The second term accounts for the expected value of detecting events, where $f(x)$ is the normalized utility of observing state x. The weighting parameters w_h and w_v control the balance between entropy reduction and detection value. The entropy term reduces redundancy: if a cell was just observed, another agent will prefer a higher-entropy cell in its field of regard. The detection value term, inspired by [15], incentivizes the agents to look at locations where the probability of observing valuable information is higher.

Algorithm 1: ABBA: Asynchronous Belief Branching for Agent i

Input: $b^e_{t_{\text{clean}}}$; σ: sync map; t_i^{sync}: sync time of agent i; π_{-i}: plans of others; H_i: Planning horizon for agent i
Output: Best open-loop plan π_i^* and its value

$\mathcal{B}[t_{\text{clean}}] \leftarrow \{(b^e_{t_{\text{clean}}}, 1)\}$;
for $t = t_{clean}$ **to** $t_i^{sync} - 1$ **do**
  $\mathcal{B}[t+1] \leftarrow$ BRANCHBELIEFS$(\mathcal{B}[t], t, \sigma, i, \pi_{-i}, \emptyset)$;
$\mathcal{C} \leftarrow$ GENERATECANDIDATES(i, H_i);
$V^* \leftarrow -\infty$, $\pi_i^* \leftarrow$ None;
foreach $\pi_i \in \mathcal{C}$ **do**
  $V \leftarrow 0$;
  foreach $(B, p) \in \mathcal{B}[t_i^{sync}]$ **do**
    $V \leftarrow V + p \cdot$ EVALSEQUENCE$(B, \pi_i, \pi_{-i}, \sigma, t_i^{\text{sync}}, H_i)$;
  if $V > V^*$ **then**
    $V^* \leftarrow V$; $\pi_i^* \leftarrow \pi_i$
return (π_i^*, V^*)

IV. ALGORITHMS

This section introduces two approaches to solve the problem in Section III. Firstly, we introduce the Asynchronous Belief Branching Algorithm (ABBA), which computes a plan by solving Equation (1). Since ABBA is not scalable, we then propose SB-ABBA, a sampling-based approximation.

A. ABBA

The idea behind ABBA, as defined in Algorithm 1, is to construct a belief tree that accounts for all possible beliefs that the central planner could hold if it received the

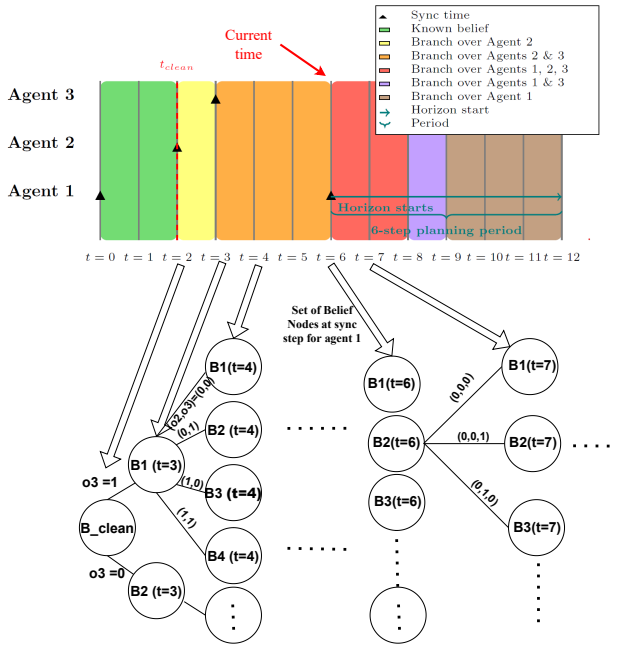

Fig. 2: Illustration of belief branching in ABBA. The top panel shows the asynchronous timeline with agent sync times, the clean time t_{clean}, and the planning horizon for agent 1. The bottom panel depicts how the global belief branches over uncertain observations (e.g., o_2, o_3), forming a tree of possible belief states. At the sync time of agent 1 ($t = 6$), the planner holds a set of weighted belief nodes that represent all possible system states consistent with past observations.

joint observations synchronously. From t_{clean} onward, each uncertain observation outcome of other agents expands the tree into separate branches, and known observations collapse parts of the environment belief into the observed values. This tree advances until reaching agent i's sync step t_i^{sync}, each branch being weighted by its probability. These probabilities are calculated by using the prior belief distribution $b^e_{t_{\text{clean}}}$, and evolving it for each branch using Equation (3). At this point, ABBA evaluates all candidate open-loop plans for agent i against this tree, expanding each node at t_i^{sync} for each possible plan π_i and accounting in each step for the other agent's given plans π_{-i} and branching on possible observation outcomes. The process of evaluating an open-loop plan appears in Algorithm 2. The selected plan is the one that maximizes the expected value across all branches. Thus, ABBA solves Equation (1) exactly.

Illustrated in Figure 2, ABBA is shown for a three-agent case. The dashed red line marks t_{clean}, after which uncertain observations from different agents create branches in the belief tree (colored intervals). Known observations are integrated into each branch. At $t = 6$, agent 1 reaches its sync point, where the planner holds a full tree of weighted beliefs. The planning horizon for agent 1 then begins, and candidate open-loop plans are evaluated against this tree. The branching continues, and the branching factor is reduced when scheduled actions for agent 2 (at t=8) and agent 3 (at t=9) expire, after which inactivity is assumed.

B. SB-ABBA

Enumerating all possible observation outcomes to build a full belief tree quickly becomes intractable. For this reason,

Algorithm 2: EVALSEQUENCE

Input: b: belief at t_i^{sync}; π_i: plan for agent i; π_{-i}: plans of others; σ: sync map; t_i^{sync}; H_i
Output: Expected discounted return for π_i

$R \leftarrow 0$; $\gamma \leftarrow$ `discount`; $\mathcal{P}[t_i^{\text{sync}}] \leftarrow \{(b, 1)\}$;
for $k = 0$ **to** $H_i - 1$ **do**
  $t \leftarrow t_i^{\text{sync}} + k$;
  $a_{i,k} \leftarrow \pi_i[k]$;
  `obs_extra` $\leftarrow \{(i, a_{i,k})\}$;
  $\mathcal{P}[t{+}1] \leftarrow$ BRANCHBELIEFS$(\mathcal{P}[t], t, \sigma, i, \pi_{-i}, \texttt{obs_extra})$;
  $R_k \leftarrow \sum_{(b',p') \in \mathcal{P}[t]} p' \cdot R(b', a_{i,k})$;
  $R \leftarrow R + \gamma^k R_k$;
return R

we adopt a sampling-based approximation inspired by classical point-based solvers for POMDPs [8]. The core idea of SB-ABBA is to replace branching with a set of particles. Each one represents a possible fully-informed system belief at the sync time t_i^{sync}. These particles are rolled forward using random actions to cover the horizon H_i. This generates a finite set of belief points $\mathcal{B}$ on which value backups are computed. Pseudocode appears in Algorithm 3.

1) Belief set construction: From the global belief $b^e_{t_{\text{clean}}}$, we generate N_{seed} independent particles by sampling the hidden observation outcomes of other agents as we roll forward to t_i^{sync}. Each particle consists of the deterministic joint clock tuple and a belief state b^e updated with known and sampled outcomes. Then, starting at t_i^{sync}, we simulate random actions for H_i steps. This yields the belief set

$$\mathcal{B} = \bigcup_{n=1}^{N_{\text{seed}}} \{(\tau^{(h)}, b^{(h,n)}) : h = 0, \ldots, H_i - 1\}, \quad (5)$$

which contains at most $N_{\text{seed}} \times H_i$ elements.

2) Value backups: For each belief point $(\tau, b) \in \mathcal{B}$, we approximate the Q-value of taking action a by Monte Carlo rollouts:

$$\widehat{Q}(\tau, b, a) = \frac{1}{N_{\text{particles}}} \sum_{n=1}^{N_{\text{particles}}} \left[r^{(n)} + \gamma V(\tau'^{(n)}, b'^{(n)}) \right], \quad (6)$$

where each sample simulates one step using the model (3), produces an immediate reward $r^{(n)}$, and transitions to $(\tau'^{(n)}, b'^{(n)})$. We use the term sweep to denote one full pass of Monte Carlo value backups over all belief points in $\mathcal{B}$.

3) Belief point matching: In SB-ABBA, each simulated step produces a successor state (τ', b') that consists of the updated clock vector and belief. Since $\mathcal{B}$ is finite, the algorithm must map (τ', b') to a representative point already in the set. We implement a three–tier lookup: (i) direct hash match on the clock vector τ', (ii) exact digest match of the belief, and (iii) symmetric KL divergence [22] search among candidates with the same clock vector. The value of the belief point with matching clock vector and the closest b', is used as the estimated value of (τ', b').

4) Open-loop plan extraction.: Once $\widehat{Q}\big((\tau^{(h)}, b), a\big)$ has been estimated, the final sequence of actions for agent i is obtained by aggregating over all belief particles that share the same clock tuple for each time step $h = 0, \ldots, H_i - 1$.

At each step, we define $\mu_\tau(h)$ as the empirical distribution of system beliefs collected during the sampling procedure. The optimal action at each step is then chosen as

$$a_h^* = \arg \max_{a \in \mathcal{A}_i(\tau^{(h)})} \mathbb{E}_{b \sim \mu_\tau(h)} \left[\widehat{Q}\big((\tau^{(h)}, b), a\big) \right], \quad (7)$$

which yields an open-loop plan $\pi_i = (a_0^*, \ldots, a_{H_i-1}^*)$ that is optimal in expectation across all sampled belief trajectories.

Algorithm 3: SB-ABBA: Sampling-Based Approximation for Agent i

Input: $b^e_{t_{\text{clean}}}$: environment belief at clean time; π_{-i}: plans of others; t_i^{sync}: sync time of agent i; H_i: horizon for agent i; N_{seed}: seeds; $N_{\text{particles}}$: rollouts; N_{sweeps}: number of value-backup passes over the sampled belief set $\mathcal{B}$ (Eq. 5)
Output: Open-loop plan π_i^*

/* 1) Build belief set $\mathcal{B}$ (Eq. 5) */
$\mathcal{B} \leftarrow \emptyset$;
for N_{seed} **times do**
  $(\tau, b) \leftarrow$ SAMPLESYSTEMSTATE$(b^e_{t_{\text{clean}}}, t_i^{\text{sync}}, \pi_{-i})$;
  for $h = 0$ **to** $H_i - 1$ **do**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\tau, b)\}$;
    $\mathcal{A}_i \leftarrow$ AVAILABLEACTIONS(i, τ);
    $a \leftarrow$ RANDOMACTION$(\mathcal{A}_i)$;
    $(\_, \tau, b) \leftarrow$ SIMULATEONESTEP(τ, b, a);

/* 2) Perform Monte Carlo backups (Eq. 6) */
for N_{sweeps} **times do**
  foreach $(\tau, b) \in \mathcal{B}$ **do**
    $\mathcal{A}_i \leftarrow$ AVAILABLEACTIONS(i, τ);
    foreach $a \in \mathcal{A}_i$ **do**
      $\widehat{Q}(\tau, b, a) \leftarrow$ average of $N_{\text{particles}}$ rollouts;
    $V(\tau, b) \leftarrow \max_a \widehat{Q}(\tau, b, a)$;

/* 3) Extract open-loop plan (Eq. 7) */
for $h = 0$ **to** $H_i - 1$ **do**
  $a_h^* \leftarrow \arg\max_{a \in \mathcal{A}_i(\tau^{(h)})} \mathbb{E}_{b \sim \mu_\tau(h)} \left[\widehat{Q}\big((\tau^{(h)}, b), a\big) \right]$;
$\pi_i^* \leftarrow (a_0^*, \ldots, a_{H_i-1}^*)$;
return π_i^*

V. EVALUATION

A. Environment Simulation

To test the proposed methods, we simulate a wildfire scenario using the assumptions defined in Section III-D. We characterize each cell in the grid by four parameters: External ignition (e.g. lightning) (λ), spontaneous ignition (β_0), contagion strength (α), and persistence (δ). These will define the local transition kernel (Equation (2)).

TABLE I: Cell types used in the experiments.

Cell Type	λ	β_0	α	δ
Immune	0.0002	0.0002	0.03	0.05
Fleeting	0.0050	0.0150	0.01	0.85
Long-lasting	0.0020	0.0020	0.01	0.99
Moderate	0.0100	0.0100	0.01	0.85
High-contagion	0.0200	0.0100	0.10	0.85

We define five cell types, shown in Table I, and we assign a type to each grid cell randomly. In each run, the environmental evolution is simulated, stored, and replayed to each planning mode. This makes for fairer comparison. Over many simulations, we obtain statistics of the performance metrics that allow us to assess the trade-offs.

In the experiments, the event state space $\mathcal{X}$ is defined as $\{\text{NO\_EVENT}, \text{EVENT\_PRESENT}\}$ and the planning horizon for each agent H_i is assumed to be the same and corresponding to their trajectory period T_i.

B. Benchmark algorithms

Given the complexity and specific structure of the described problem, it is not trivial to choose benchmark planning methods to test the proposed solutions. To the best of our knowledge, no algorithm exists tailored to these specifics. Therefore, we evaluate our approach against a diverse set of baselines, including several strong heuristics as well as two open-loop planners adapted to operate over the belief state.

a) Oracle: Oracle is a rule-based planner that assumes perfect knowledge of where and when events occur and instant communication with all agents, so it always chooses the best possible sensing actions to optimize event detection and NDD. It serves as an optimistic upper bound.

b) Random: Agents choose uniformly among the available sensing actions at each phase. This strategy ignores the belief and serves as a lower bound.

c) MPOMDP Open-Loop Planner (MOLP): An exact MPOMDP solver. At each replanning step, i.e., when an agent synchronizes with the central planner, it enumerates every possible joint action sequence and selects the one with the highest expected discounted reward over the next period. It assumes simultaneous communication with all agents, hence it is "cheating" with respect to ABBA and SB-ABBA. This allows us to assess whether our algorithms achieve comparable performance under the communication constraints they explicitly model.

d) KL-OLOP: Per-agent open-loop Monte Carlo tree search with KL-divergence exploration. We adapt it to operate on belief states and it serves as an open-loop analogue of POMCP. Re-planning is done for one period each contact. For the original implementation, see [23].

e) Greedy: At each synchronization, the central planner selects sequences of actions maximizing the immediate expected reward from the current belief. It does not reason about future outcomes or coordination with other agents.

f) Prior-based: Actions are sampled from a static prior distribution over event likelihood derived from the spread model parameters. The policy does not update decisions after observations and ignores coordination.

g) Sweep: Agents follow predetermined coverage patterns aligned with their periodic trajectories, e.g., left-to-right or right-to-left sweeps across the grid. The purpose is to perform systematic coverage of the grid, but without reacting to new information.

C. Results

1) 4×3 Case: In this case study, we simulate a small grid with 4 rows and 3 columns. Two agents are deployed, and their field of regard (see Definition 2) at each phase is

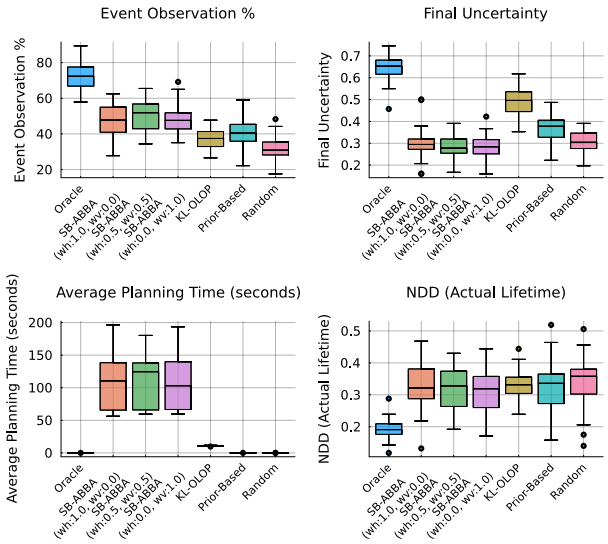

Fig. 3: Results for the 5×5 case averaged over 30 simulations of 100 timesteps each. SB-ABBA is shown under three reward weight configurations: $(w_h{=}1, w_v{=}0)$, $(w_h{=}0.5, w_v{=}0.5)$, and $(w_h{=}0, w_v{=}1)$.

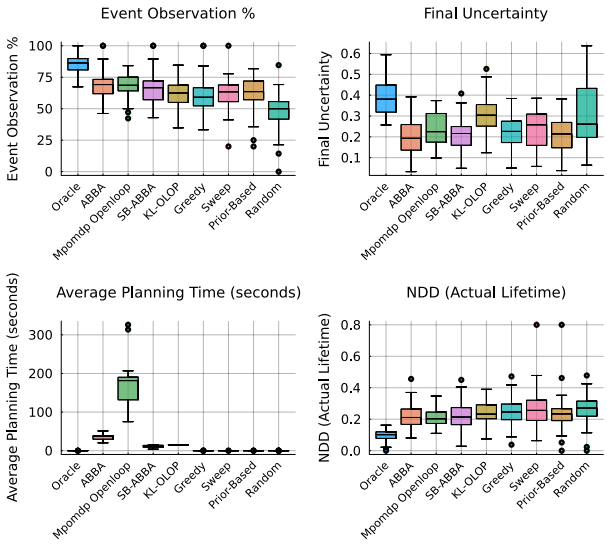

Fig. 4: Results for the 4×3 case for 30 simulations of 100 timesteps.

their current row, but their sensor field of view covers only one cell within that row. The agents cycle upward through the rows and reset at the bottom, so the trajectory period is 4. We also set the planning horizon to this period, since agents revisit the same cells every 4 steps. SB-ABBA is configured with $N_{seed} = 30$, $N_{particles} = 64$, $N_{sweeps} = 50$. Boxplots of the results are shown in Figure 4, and these are obtained after averaging over 30 independent simulations of 100 timesteps each. Discarding Oracle and MOLP, which use privileged information, it is clear that ABBA is able to beat the other benchmarks in both NDD and final uncertainty; but this comes at a high price in terms of planning time, with around 34 seconds per plan on average. On the other hand, SB-ABBA achieves similar performance, but using much less time, around 9 seconds per plan. Among the heuristics, the best performing one is the prior based; its performance is still below both ABBA's and SB-ABBA's. Greedy falls well below these methods, which shows that coordination and considering future rewards is highly important in this problem. KL-OLOP achieves good performance, but still ranks below ABBA and SB-ABBA. A remarkable insight from these results is that both ABBA and SB-ABBA are very close in performance to MOLP in EOP, NDD, and uncertainty, and have significantly lower computing time.

2) 5×5 Case: This case study consists of a 5×5 grid with two agents deployed. Their field of regard at each phase is a cross centered on their current cell, while the sensor field of view covers only one cell within that cross. The agents follow fixed periodic trajectories of length 10, and we set the planning horizon equal to this period. SB-ABBA is configured as in the 4×3 case. Figure 3 shows boxplots averaged over 30 independent simulations of 100 timesteps each. We tested SB-ABBA with three different weight configurations for the reward function: $(w_h{=}1, w_v{=}0)$, $(w_h{=}0.5, w_v{=}0.5)$, and $(w_h{=}0, w_v{=}1)$. As expected, entropy-heavy weights encourage broader coverage and lower redundancy, while detection-heavy weights prioritize event locations but leave higher residual uncertainty. The balanced configuration achieves a compromise between the two. Across all metrics, SB-ABBA outperforms the heuristic baselines and KL-OLOP, with prior-based again the strongest after SB-ABBA. Planning time remains moderate compared to ABBA, which was unfeasible for this problem scale, confirming that SB-ABBA scales better while maintaining competitive performance.

3) 9×9 Case: To assess the scalability of SB-ABBA to larger problem instances, we applied the algorithm to a 9×9 grid with 4 agents and a planning horizon of 12 as shown in Figure 1 (left). Four UAVs follow periodic circular trajectories to monitor wildfires and receive new instructions from the planner when they fly over the ground station in the center of the grid. Boxplots are shown in Figure 5. SB-ABBA's configuration is identical to the previous cases.

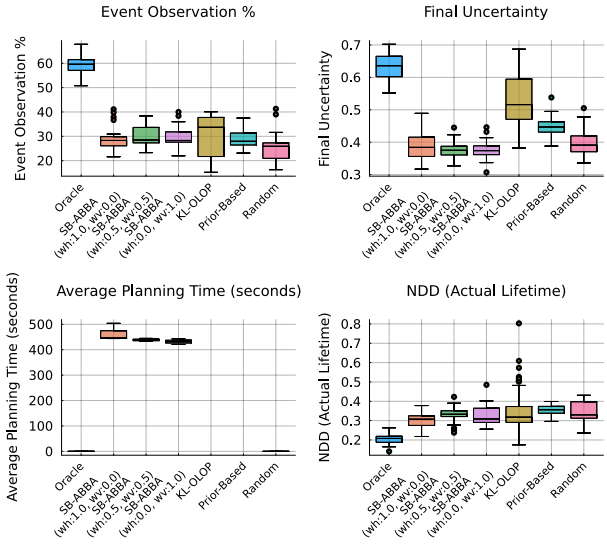

Fig. 5: Results for the 9×9 case for 30 simulations of 100 timesteps.

VI. CONCLUSIONS AND FUTURE WORK

We presented MA-APPOMDP, a formal framework that extends Dec-POMDPs and MPOMDPs to address a specific

TABLE II: Summary of average performance across case studies. Oracle and MOLP use privileged information so they act as upper bounds.

Grid	Method	Obs.%	Unc.	NDD	Time (s)
	Oracle	86.4	0.383	0.098	0.0
	MOLP	69.1	0.235	0.212	169.0
	ABBA	**68.4**	**0.193**	**0.214**	34.7
	SB-ABBA	66.4	0.206	0.219	10.2
4×3	KL-OLOP	62.9	0.302	0.246	10.8
	Prior	63.5	0.209	0.231	0.004
	Sweep	61.8	0.236	0.263	0.006
	Greedy	59.7	0.220	0.249	0.001
	Random	48.4	0.305	0.264	**0.0**
	Oracle	72.3	0.644	0.193	0.0
	SB-ABBA (0,1)	48.2	**0.283**	**0.311**	109.0
	SB-ABBA (0.5,0.5)	**50.1**	**0.283**	0.321	107.1
5×5	SB-ABBA (1,0)	47.6	0.299	0.324	104.8
	KL-OLOP (0.5 0.5)	37.5	0.493	0.332	15.6
	Prior	40.8	0.364	0.328	0.003
	Random	31.5	0.305	0.337	**0.0**
	Oracle	59.8	0.633	0.204	0.0
	SB-ABBA (0,1)	29.1	0.387	**0.306**	464.6
	SB-ABBA (0.5,0.5)	**30.2**	0.377	0.332	438.5
9×9	SB-ABBA (1,0)	30.0	**0.376**	0.328	431.8
	KL-OLOP (0.5, 0.5)	29.9	0.527	0.354	24.4
	Prior	28.8	0.446	0.353	0.005
	Random	25.6	0.399	0.345	**0.0**

subset of multi-agent planning problems where agents follow periodic trajectories and communicate asynchronously with a central planner. Along with this problem formulation, two planning algorithms have been presented: ABBA and SB-ABBA. In all cases, as shown in Table II, they achieved higher event coverage and the lowest NDD. The improvement is about 8% for EOP in the 4×3 case and 23% in the 5×5 case, with residual uncertainty reduced by 8–22% and NDD improving by 5–7%. In the 9×9 case, coverage gains are smaller (about 5%), but residual uncertainty decreases by nearly 16% and NDD improves by about 6%. The results show two main effects. First, our coordination-aware planning methods consistently outperform individual planning strategies such as KL-OLOP, greedy policies and other heuristics. Second, ABBA and SB-ABBA achieve performance very close to MOLP, despite operating under asynchronous communication constraints. This indicates that reasoning over the joint multi-agent belief evolution can effectively approximate centralized coordination in asynchronous communication settings. The main limitation of the approach is the scalability of SB-ABBA, since planning time increases significantly in the 9×9 case, although it remains substantially more tractable than ABBA.

Next steps include scaling to larger problems, implementing compact reactive policies (particularly when agent updates are firmly tied to sync times) that allow the collection of data needed to learn transition and reward models, and developing shared models between agents. Another line of work is to explore how to model communication itself as a decision, so the planner can adapt the sync schedule and content of the communications based on the needs of agents. Finally, advancing theoretical analysis of SB-ABBA, and empirical testing on actual UAV or satellite systems will be critical for providing value to practitioners.

Acknowledgment Supported by ONR Award #N00014-22-1-2476. A portion of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under NASA contract (80NM0018D0004). ChatGPT was used for grammar enhancement.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998.

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

[3] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small UAVs," in *Proceedings of the American Control Conference*, 2005, pp. 3530–3535.

[4] J. Scherer and B. Rinner, "Persistent multi-UAV surveillance with data latency constraints," *arXiv preprint arXiv:1907.01205*, 2019.

[5] J. P. Wilhelm, G. S. Clem, and G. M. Eberhart, "Direct entry minimal path UAV loitering path planning," *Aerospace*, vol. 4, no. 2, p. 23, 2017.

[6] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, vol. 6, no. 5, pp. 367–381, 2002.

[7] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile Earth observation satellite scheduling over 20 years: Formulations, methods, and future directions," *IEEE Syst. J*, vol. 15, no. 3, pp. 3881–3892, 2020.

[8] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003, pp. 1025–1032.

[9] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[10] W. S. Lovejoy, "A survey of algorithmic methods for partially observed Markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 47–65, 1991.

[11] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[12] S. Seuken and S. Zilberstein, "Memory-Bounded Dynamic Programming for DEC-POMDPs," in *Proc. Joint Conf. on Artificial Intelligence (IJCAI)*, 2007, pp. 2009–2015.

[13] M. Merlin, N. Parikh, E. Rosen, and G. Konidaris, "Locally observable Markov decision processes," in *ICRA Workshop on Perception, Action, Learning*, 2020.

[14] L. Dressel and M. Kochenderfer, "Efficient Decision-Theoretic Target Localization," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 27, no. 1, pp. 70–78, June 2017.

[15] X. Zhou, W. Wang, T. Wang, M. Li, and F. Zhong, "Online planning for multiagent situational information gathering in the Markov environment," *IEEE Syst. J*, vol. 14, no. 2, pp. 1798–1809, 2019.

[16] M. Lauri, E. Heinänen, and S. Frintrop, "Multi-robot active information gathering with periodic communication," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 851–856.

[17] X. Zhou, W. Wang, T. Wang, X. Li, and T. Jing, "Continuous patrolling in uncertain environment with the UAV swarm," *PLOS ONE*, vol. 13, no. 8, p. e0202328, 2018.

[18] G. Rabideau, D. Tran, S. Chien, B. Cichy, R. Sherwood, D. Mandl, S. Frye, S. Shulman, J. Szwaczkowski, D. Boyer, *et al.*, "Mission operations of earth observing-1 with onboard autonomy," in *IEEE International Conference on Space Mission Challenges for Information Technology*, 2006, p. 373.

[19] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," Ph.D. dissertation, University of California, Berkeley, Computer Science Division, 2002.

[20] N. Grieshop and C. K. Wikle, "Data-driven modeling of wildfire spread with stochastic cellular automata and latent spatio-temporal dynamics," *Spatial Statistics*, vol. 59, p. 100794, 2024.

[21] C. Díaz-Avalos and P. Juan, "Modeling the spatial evolution wildfires using random spread process," *Environmetrics*, vol. 33, no. 8, p. e2774, 2022.

[22] H. Chan and A. Darwiche, "A distance measure for bounding probabilistic belief change," *International Journal of Approximate Reasoning*, vol. 38, no. 2, pp. 149–174, 2005.

[23] E. Leurent and O.-A. Maillard, "Practical open-loop optimistic planning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 69–85.