A Hierarchical Coordination Framework for Joint Perception-Action Tasks in Composite Robot Teams

Esmaeil Seraj, *Student Member, IEEE,* Letian Chen, *Student Member, IEEE,* and Matthew C. Gombolay, *Member, IEEE,*

***Abstract*—We propose a collaborative planning and control algorithm to enhance cooperation for composite teams of autonomous robots in dynamic environments. Composite robot teams are groups of agents that perform different tasks according to their respective capabilities in order to accomplish an overarching mission. Examples of such teams include groups of perception agents (can only sense) and action agents (can only manipulate) working together to perform disaster response tasks. Coordinating robots in a composite team is a challenging problem due to the heterogeneity in the robots' characteristics and their tasks. Here, we propose a coordination framework for composite robot teams. The proposed framework consists of two hierarchical modules: (1) a Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) algorithm in Multi-Agent Partially Observable Semi-Markov Decision Process (MA-POSMDP) as the high-level decision-making module to enable perception agents to learn to surveil in an environment with an unknown number of dynamic targets and (2) a low-level coordinated control and planning module that ensures probabilistically-guaranteed support for action agents. Simulation and physical robot implementations of our algorithms on a multi-agent robot testbed demonstrated the efficacy and feasibility of our coordination framework by reducing the overall operation times in a benchmark wildfire-fighting case-study.**

***Index Terms*—Perception-Action Coordination, Semi-MDP, RL Exploration, Hierarchical Control.**

I. Introduction

MULTI-ROBOT teams are able to execute time-sensitive, complex missions by cooperatively leveraging their unique capabilities and design [1, 2]. Heterogeneity in robots' design characteristics and their roles are introduced to (1) leverage the relative merits of different agents and their capabilities [1] and, (2) deal with the dynamic and unpredictable nature of the real-world for which designing homogeneous, versatile robot teams that can effectively adjust to all circumstances is difficult and costly [3]. This introduction entails the formation of *composite teams* of heterogeneous, co-dependent agents, in which different robots speciate in different parts of a compound or complex task (see Korsah et al. [4] for a complete taxonomy of tasks in multi-robot teams).

An important instance of composite robot-teams is the collaboration between *perception (sensing) agents* and *action (manipulator) agents* [5, 6]. In a perception-action composite robot team, to complete a mission, perception robots are first, tasked to explore an unknown environment to find an initial set of dynamic targets and then, *exploit* those targets. Estimated target-states are required by action agents to perform a specific manipulation on those targets. In an example aerial wildfire fighting task, perception UAVs first explore the environment to find fire spots. Once key fire spots are identified, the perception agent will loiter to gain a higher quality state estimate of the fire targets for an action UAV to be able to efficiently douse each fire, thereby *exploiting* the targets. In this paper, we make the common assumption that perception robots are only capable of sensing while action robots are only capable of manipulating. We note that in dynamic environments, the described exploration versus exploitation trade-off for perception robots needs to be decided in a *restless* manner [7], in which targets' states, such as position and velocity, evolve over time regardless of agents' actions. Applications of such composite robot teams are numerous in surveillance and disaster response [6, 8, 9], search and rescue [10, 11], manufacturing [12], and border patrol [13].

Previous studies tackle various aspects of heterogeneous multi-robot teaming by considering coordination strategies [14, 15, 16, 17], task allocation [18, 19, 20, 21, 22], and path-planning and control [2, 23, 24, 21, 25, 26]. The problem of coordination and collaborative planning between perception and action agents has been of keen interest to the wireless communication research community, referring to the problem as *Wireless Sensor and Actor Networks* [6, 9]. While most of this prior work studies static environments (e.g., [27, 8, 9, 12, 10]), this assumption does not hold true in many environments of interest which are dynamic (e.g., including numerous, moving targets). Such dynamic environments have attributes of both a Partially Observable Markov Decision Process (POMDP) and a Restless Bandit Problem [28]. Unfortunately, traditional Reinforcement Learning (RL) formulations lack the scalability and adaptability with respect to domain shift in order to tackle real-world problems. Moreover, most of the proposed nonlearning-based approaches (e.g., mixed-integer linear/nonlinear program such as [15]) fail to handle the large-scale, dynamic, and stochastic nature of these problems.

Efficient planning and coordination of robots with different traits in a composite team while accounting for their collaborative behavior through specific capabilities and limitations is of significant importance [6, 5, 1]. This coordination becomes more challenging when the dynamicity of the environment needs to be taken into account [12]. Coordinating composite

E. Seraj, L. Chen and M. C. Gombolay are with the Institute for Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, 30332 USA. (*Corresponding Author: Esmaeil Seraj*) e-mail: eseraj3@gatech.edu, letian.chen@gatech.edu and matthew.gombolay@cc.gatech.edu.

Manuscript received May 30, 2020. Accepted June 17, 2021.

This work was supported by the Office of Naval Research (ONR) under grant N00014-19-1-2076 and the Naval Research Laboratory (NRL) under grant N00173-21-1-G009.

teams consists of two key components: (1) developing a high-level decision-making module to balance the exploration and exploitation of dynamic targets and (2) designing a low-level control module with inclusive policies to individualize the inter-robot interactions for improving cooperation resiliency and overall performance. In other words, the high-level module is responsible for deciding on "*what to do?*" and "*where to go?*", while the low-level controller focuses on resolving "*how to do it?*" and "*how to get there?*".

Our work overcomes the limitations of prior work by proposing a novel hierarchical approach (Fig. 1) to jointly tackle the (1) high-level decision-making and (2) the low-level coordinated control problems for heterogeneous teams of autonomous robots consisting of perception agents and action agents. The high-level decision-making module is centralized and is responsible for effectively balancing the exploration vs. exploitation trade-off for perception agents. The low-level module is a fully distributed coordinated controller that guides the perception agents to generate action-agent-aware, performance-guaranteed trajectories for action agents. To enable our framework, we propose a novel RL algorithm leveraging the State-Action-Reward-State-Action (SARSA) algorithm within a Multi-Agent Partially Observable Semi Markov Decision Process (MA-POSMDP); our approach enables us to learn a policy for perception agents that addresses the high-level exploration versus exploitation dilemma in an unknown, dynamic environment in support of action agents. We consider an unknown, variable number of moving targets with known motion models to be explored by the perception robots, while also tasking them to exploit the found targets to (1) extract (estimate) the necessary state-information, e.g. dynamics such as instantaneous position and velocity, (2) generate a feasible trajectory for action robots, and finally (3) task action agents to manipulate in such a way that their performance can be guaranteed. We augment a probabilistic-bound from prior work with perception-only teams [24] to now capture important dynamics of perception-action teams vis-à-vis the maximum tracking error allowed for action actions to service targets denoted by perception agents. We leverage these upper-bound times into the high-level decision-making by explicitly modeling action durations, as per a Semi Markov Decision Process [29]. We design an attribute-based robot-interaction scheme between perception and action robots to increase the coordination resiliency and efficiency.

Our comprehensive, hierarchical framework includes a learning-based solution for MA-POSMDP, online Extended Kalman Filter (EKF) based target state estimation and uncertainty prediction at the higher (i.e. perception) level as well as a probabilistic upper-bound time setting for target service through teaming the perception and action robots at the lower level. Our approach enables interactive solutions for the high-level decision-making to explore and scan an unknown environment as well as the low-level control to coordinate co-dependent agents' actions jointly with low-level performance guarantees. To our knowledge, this challenging set of problems and the development of such a coherent system have never been approached as a whole before; our paper is the first.

We empirically validate our framework in simulation and physical-robot implementation on the Robotarium multi-agent robot testbed [30]. Our evaluations demonstrate the efficacy and feasibility of our coordination framework by reducing the overall operation times in a benchmark wildfire-fighting case-study. We choose the application of wildfire fighting via heterogeneous, autonomous Unmanned Aerial Vehicles (UAV) as the case-study and motivating application for this work.

Contributions – The primary contributions of our work are:

1) Formulating a novel algorithm Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) to learn a high-level decision making policy in an unknown, dynamic environment modeled by a Multi-Agent Partial Observable Semi Markov Decision Process (MA-POSMDP).
2) Deriving an analytical upper-bound on tracking error that enables perception and action agents to cooperatively determine whether action agents can provide a probabilistically-guaranteed service for a set of manipulation tasks given the state information received from perception agents for those tasks.
3) Proposing an attribute-based, individualized, coordination scheme between perception and action agents for an efficient, coordinated routing problem to set a state-of-the-art, outperforming our baselines in performance.

II. Motivating Application: Aerial Wildfire Fighting

We motivate the coordination of heterogeneous composite robot teams with perception and action agents in the problem of aerial wildfire fighting [5]. Firefighters require online and accurate information about the wildfire states (e.g., position, propagation velocity and direction, etc.) to plan for using large, limited, and high-cost aerial firefighting equipment such as Airtankers, Water Scoopers and Smokejumper aircraft [2, 17]. To gather this required information, however, other smaller, low-cost, and low-power devices such as multi-rotor quadcopters are more desirable due to their agility and cost-efficiency. These small information-gathering UAVs can perform as perception robots, while the larger resource-rich aircraft, equipped with fire retardant, can be considered as action robots.

Our objective is three-fold: (1) given an unknown dynamic wildfire environment, balance the time spent on exploring new fire-areas (e.g., firespots) and exploiting the ones already found for extracting state-information such as fire's location and spread velocity, (2) efficiently communicate the exploited fire-areas between quadcopters (perception agent) and Airtankers (action agents) such that generated firefighting paths are feasible by the assigned action UAVs and (3) provide a probabilistic guarantee for the composite firefighting UAV team's ability to coordinate their activities.

A. FARSITE: Wildfire Propagation Model

We leverage the simplified Fire Area Simulator (FARSITE) wildfire propagation model [31, 32], where q_t^i indicates how firespot i propagates according to Eq. 1 along the X-Y coordinates and thus, q is a 2-dimensional vector of size 1×2.

$$q_t^i = q_{t-1}^i + \dot{q}_{t-1}^i \delta t \tag{1}$$

The above equation represents the fire motion model in which, $\dot{q}_t^i = dq_t^i/dt$ is the motion dynamics and identifies a fire's growth rate (i.e., fire propagation velocity) and is a function of fuel and vegetation coefficient (R_t), wind speed (U_t), and wind azimuth (θ_t). The first-order firespot dynamics, $\dot{q}_t$, can be estimated for each propagating spot by Eq. 2, where $\mathcal{D} = \sin(\theta_t)$ and $\mathcal{D} = \cos(\theta_t)$ along *X* and *Y* axes, respectively [31].

$$\dot{q}_t = C(R_t, U_t)\mathcal{D}(\theta_t) \tag{2}$$

In Eq. 2, the spread rate $C(R_t, U_t)$ can be calculated as $C(R_t, U_t) = R_t\left(1 - \frac{LB(U_t)}{LB(U_t)+\sqrt{GB(U_t)}}\right)$, in which $LB(U_t) = 0.936e^{0.256U_t} + 0.461e^{-0.154U_t} - 0.397$ and $GB(U_t) = LB(U_t)^2 - 1$. While fire is propagating, we assume each firespot radiates heat according to a Gaussian distribution and the intensities of adjacent points are linearly summed if the two points are within a predefined radiation range, which is in accordance with prior studies [32]. Furthermore, due to the fuel exhaustion, we model the intensity decay of a fire spot during its ignition time δt_q, as a dynamic exponential decay rate λ over time, as $I^q_{t+\delta t} = I^q_t\left(e^{-\lambda\frac{\delta t_q}{R_t}}\right)$. In this equation, I^q_t is the heat intensity of firespot q at time t and is calculated according to the intensity model proposed by [33], in which $I^q_t = 259.833\left(\frac{h^q_t}{\cos(\alpha^q_t)}\right)^{2.174}$. In this equation, h^q_t and α^q_t are flame height (meters) and tilt angle with respect to vertical horizon line (degrees) for each firespot and the intensity I^q_t is measured in kilo-watts per meter, $[\frac{KW}{m}]$ units.

Finally, when a firefighting UAV drops the extinguisher fluid over an area of fire, we cut the fire intensities of the respective fire spots according to a predefined extinguisher fluid coefficient. A fire point is pruned from the fire-map if its intensity falls below a threshold value, leaving a burnt spot on the terrain map which cannot catch fire anymore.

III. Problem Statement and Formulation

A. Road Map

We first provide a high-level perspective on contributions and content provided in the paper as a road-map listed as follows:

Step 1) We propose a novel formalization to describe the high-level decision making problem, namely Multi-Agent Partially Observable Semi Markov Decision Process (MA-POSMDP) and develop a novel variant of SARSA called MA-SARTSA as our learning-based solution (Section IV). In this step, perception agents need to select among exploration (find new targets), exploitation (execute tasks with action agents by passing extracted target state information) or revisiting a previous target which has been acted upon via an action agent (re-examine targets). Re-examining a target is also dependent on the respective action agent's upper-bound service time ($\mathcal{T}_{\mathcal{UB}}$) computed in Step 5 (see Fig. 1).

Step 2) Considering the high-level exploitation and revisiting actions from Step 1, we begin the low-level planning and control by tackling the online target state estimation through EKF for perception agents (Section V-A). Here, we record measurement uncertainties to derive a tracking error upper-bound in Step 4.

Step 3) We provide our low-level scanning framework to individualize the interaction between perception and action robots to improve resiliency and cooperation efficiency (Section V-B). The objective of this step is two-fold: (A) to generate an action-agent-aware trajectory to account for the heterogeneity of perception and action robots and (B) make the probabilistic upper-bound service times calculated in Step 5 more accurate.

Step 4) Given EKF's covariance matrices from Step 2, we propose our uncertainty-based analytical tracking error upper-bound ($\text{TEB}^q_{\mathcal{T}_{\mathcal{UB}}}$) which performs as a quantitative probabilistic performance guarantee for action agents which are incapable of sensing the dynamic targets. $\text{TEB}^q_{\mathcal{T}_{\mathcal{UB}}}$ will then be used with the upper-bound service times for action agents, $\mathcal{T}_{\mathcal{UB}}$, (Section V-C) in Step 5 to generate a set of target waypoints for these agents.

Step 5) We derive a set of probabilistic upper-bound service times $\mathcal{T}_{\mathcal{UB}}$ for action agents (Section V-D) to: (A) provide an upper-bound on the number of assigned tasks to a particular action agent (incapable of sensing) and guaranteeing that it will not miss the moving targets and (B) generate a time quantity as an input for perception agents' decision making in Step 1 (see Fig. 1).

B. Problem Description and Formulation

To formulate the described dynamic optimization problem, consider a total of N_T dynamic targets with state-space $\left(s_t^{T_1}, \cdots, s_t^{T_{N_T}}\right) \in \{S_t^T\}^{N_T}$ (T represents target) and a known dynamics model, $\mathcal{M}_t$ (e.g., motion model introduced in Eq. 1), but unknown parameter settings for the model (e.g., velocity as introduced in Eq. 2). Further, consider a total of $N = N_P + N_M$ robots including N_P perception agents (i.e., sensing UAVs) [1] with state-space $\left(s_t^{P_1}, \cdots, s_t^{P_{N_P}}\right) \in \{S_t^P\}^{N_P}$ (P represents perception) and N_M action agents (i.e., manipulator UAVs) with state-space $\left(s_t^{M_1}, \cdots, s_t^{M_{N_M}}\right) \in \{S_t^M\}^{N_M}$ (M represents manipulation). Each robot state-vector is defined as $s_t^{P_i} \in \{S_t^P\}^{N_P}$ for perception (P) agents and $s_t^{M_i} \in \{S_t^M\}^{N_M}$ for manipulator (M) agents and contains robots' position, velocity and trait information. Similarly, each target state-vector, $s_t^{T_i} \in \{S_t^T\}^{N_T}$, contains each target's position and velocity information alongside any other related, application-dependent state variable, such as R_t, U_t and θ_t in the fire propagation model for the application of aerial wildfire fighting.

We note due to the dynamicity of the environment (e.g., constantly state-changing targets), the problem resembles *restless decision making* problem [27]. In our restless decision-making problem, assuming $N_P < N_T$, at each time t, there will always be a state-changing target that is not covered by any perception agent. Here, regardless of the collective decisions of the perception-action team, the states of all observed or unobserved targets are constantly evolving through time. Conventionally, this restless problem would normally result in receiving two local *active* and *passive* rewards (i.e., r_t^{active} and $r_t^{passive}$ respectively), by the perception agent for

[1]Terms "perception agent/robot" vs. "sensing UAV" and "action agent/robot" vs. "manipulator UAV" are used interchangeably throughout.

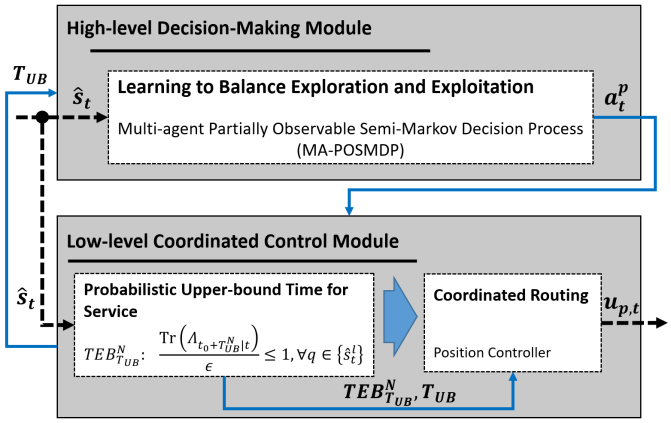


Figure 1: This figure depicts the proposed hierarchical coordination framework. Dashed links pass control input u_t to and receive state information $\hat{q}_t$ from robots. The higher-level leverages service time upper-bounds $\mathcal{T}_{\mathcal{UB}}$ from low-level controller when making decisions to generate actions.

both observed and unobserved targets [34]. However, here we combine these two rewards into a single overarching reward to encourage cooperation among perception agents (see Section IV-D) through team-based rewards.

This problem can be considered as a variant of POMDP. Accordingly, our objective is to find an optimal policy, π^*, over all admissible policies in set $\pi \in \Pi$, that maximizes the total expected, time-discounted reward accumulated by all perception agents over an infinite horizon, as in Eq. 3. We provide detailed problem formulation and solution to this problem in Section IV.

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r_t^{P_1} + \cdots + r_t^{P_{N_P}} \right) \right] \tag{3}$$

For perception agent, P_i, if an action, $a_t^{P_i}$, at time t, is exploitation, it means the robot must extract the state-information from an already-found target point $s_t^{T_i} \in S_t^T$ to be passed to action agent M_j. Perception agents explore the environment and exploit the discovered targets to collectively generate a unified list of targets' estimated states, $\mathcal{L}_t = \{\hat{s}_t^1, \cdots, \hat{s}_t^{l(t)}\}$, in which $\hat{s}_t$ are targets' estimated state vectors. The length $l(t)$ changes with time, since exploring targets adds more exploitation options to the list and exploiting targets will remove targets from the list. We note that this variable-length state vector leads to a non-trivial dynamic-size state- and action-space representation problem in a learning-based decision making approach. We tackle this problem and propose a solution in Sections IV-C - IV-E.

To increase the efficiency of the composite robot team, we generalize the coordination problem such that action robots are capable of manipulating on more than one target during a single deployment. For example, an Airtanker UAV can fight the wildfire in more than just one grid spot. Accordingly, each vector in the list $\mathcal{L}_t$ is a set of estimated target states, $\hat{s}_t^l = \{\hat{q}_t^{m_1}, \cdots, \hat{q}_t^{m_c}\}$, in which m is the index of the manipulator agent and c is length of the waypoint list $\hat{s}_t^l$ sent to robot m. To determine the length, c, different factors such as targets' state transition model, $\mathcal{M}_t$ (Eq. 1), action agent's motion/flight dynamics (e.g., maximum velocity, v_{max}, and turning bank, ω_{max}), and battery restrictions, need to be taken into account. We note that, due to dynamicity of the targets, simply passing the current coordinates of a sensed target from a perception agent to an action agent does not work. Accordingly, a probabilistic framework based on both target's motion model and action agent's dynamics is needed.

In the context of dynamic-target tracking, we need to make a high-level decision on whether to explore new targets or "exploit" known targets. Exploration implies searching the environment for new targets, while exploitation means examining an already found target to extract (estimate) necessary information for action agents and carry out the task by passing state-estimates to an action agent. When a perception robot, i, with position, $p_t^{P_i} \in s_t^{P_i}$, exploits an already-found target, initially, the closest action robot, j, with position, $p_t^{M_j} \in s_t^{M_j}$, is assigned to the task. To ensure the assigned action robot with current position, $p_t^{M_j}$, and maximum linear velocity, $v_{max}^{M_j}$, does not miss the moving target, τ, with estimated position, $\hat{q}_t^\tau$, and velocity, $\dot{\hat{q}}_t^\tau$, where $\left(\hat{q}_t^\tau, \dot{\hat{q}}_t^\tau\right) \in \hat{s}_t^\tau \in S_t^T$, we derive a measurement-uncertainty-based analytical probabilistic upper-bound time $\mathcal{T}_{\mathcal{UB}}$ (see Fig. 1), which determines the upper-bound time required for the selected action agent to reach a close-enough proximity of the location of the sensed target. To this end, the new location, $\hat{q}_{t+\mathcal{T}_{\mathcal{UB}}}^\tau$, that the moving target, $\hat{q}_t^\tau$, will move to, while the manipulator agent is on its way, needs to be estimated. To estimate $\hat{q}_{t+\mathcal{T}_{\mathcal{UB}}}$, perception agents leverage EKF's multi-step prediction framework to propagate the detected point for $\mathcal{T}_{\mathcal{UB}}$ time into the future based on the target's motion model, $\mathcal{M}_t$, (e.g., fire propagation model in Eq. 1). See Sections V-A and V-C for details of this process.

Sensing robots continue to add propagated, explored targets to the set $\hat{s}_t^l = \{\hat{q}_{t+\mathcal{T}_{\mathcal{UB}}}^{m_1}, \cdots, \hat{q}_{t+\mathcal{T}_{\mathcal{UB}}}^{m_c}\} \in \mathcal{L}_t$, to be sent to a selected action robot. This process is performed through executing a feasibility test by comparing $\mathcal{T}_{\mathcal{UB}}$ to a maximum time allowable for each target track to propagate before the measurement uncertainty residual exceeds an acceptable predefined bound. The feasibility test, which we refer to as tracking-error bound (TEB) check, is preformed while specifically accounting for motion and battery restrictions of the assigned action manipulator robot, to jointly determine the length, c, of waypoint set, $\hat{s}_t^l$, and to generate executable trajectories within regions *reachable* by the action agent. See Sections V-B and V-D for details.

If the TEB check is satisfied for a target point $\hat{q}_{t+\mathcal{T}_{\mathcal{UB}}}$, the target is added to the waypoint set, $\hat{s}_t^l$, by perception agent and the robot then moves to execute the next action decision, a_t^{Pi}, made by the high-level decision-maker. The TEB check is performed continuously at each time point for all the points on the waypoint list. This process continues until the first time the TEB check fails for a point or the overall travel time to visit all the nodes on the waypoint set reaches the assigned manipulator agent's battery limit. Now, the generated set of waypoints, $\hat{s}_t^l$, will be sent to the action agent. If the TEB check fails for the first point in a set, the perception agent rejects the assigned action agent and recruits another closer and/or faster robot to execute the task, if available (see Sections V-C and V-D for details). For readers' convenience, we provide Table I listing

Table I: Summary of key nomenclature used in our paper.

Notation	Domain	Definition and Properties
N_T	$\mathbb{N}^+$	Total number of targets
N_P	$\mathbb{N}^+$	Total number of perception agents
N_M	$\mathbb{N}^+$	Total number of manipulator agents
$\hat{}$	$-$	Accent used for estimated variables
q^τ	$\mathbb{R}^{1\times 2}$	Location of τ-th firespot
p^{P_i}	$\mathbb{R}^{1\times 3}$	Position of i-th perception agent
p^{M_j}	$\mathbb{R}^{1\times 3}$	Position of j-th action agent
$\mathcal{L}_t$	$\mathbb{R}^{1\times l(t)}$	List of estimated waypoints at time t
$l(t)$	$\mathbb{N}^+$	Length of $\mathcal{L}_t$ at time t
$\hat{s}^l$	$\mathbb{R}^{1\times c}$	Estimated target state list; l-th element of $\mathcal{L}_t$
c	$\mathbb{N}^+$	Length of $\hat{s}^l$ at time t
$\mathcal{S}$	$\mathbb{R}^{w_1\times w_2\times k}$	State space; $w_1 \times w_2$ is the world size
$\mathcal{O}$	$\mathbb{N}^{w_1\times w_2\times k}$	Observation space; k is number of features
a	$\{0,1\}^{w_1\times w_2}$	Perception agent action representation
R	$\mathbb{R}$	Total accumulated discounted reward
$\mathcal{T}_{\mathcal{UB}}$	$\mathbb{R}$	Upper-bound service time for an action agent
TEB^q	$\mathbb{R}$	Tracking-error Bound for target q
$\mathcal{E}$	$\mathbb{R}$	An acceptable threshold for TEB

the key variables used throughout the article. The following sections detail the modules and steps in our framework.

IV. High-level Decision Making

As described in Section III, our objective in the high-level decision-making module is to automatically balance perception agents' effort in exploring the environment and exploiting found targets. To this end, we propose a learning framework termed Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) learning to enable high-level decision making in a Multi-Agent, Partially Observable Semi-MDP. We note that while high-level decision-making directly governs the perception agents' actions, action agents indirectly contribute to this decision-making process because (1) action agents' dynamics and motion characteristics are explicitly considered by perception agents when generating the performance-guaranteed task trajectories (e.g., waypoint set of targets) for those action agents and (2) action agents execute their tasks (i.e., manipulate the target) in an exploitation action[2].

A. Ground-Truth Environment Model

We assume that only the targets' motion model is known and do not assume any other prior information about targets, i.e. the number of targets or the parameters of targets' motion model. Thus, we are learning in a partially observable environment in which the state transition and the state transition times (as in Semi-MDP) may depend on unobserved variables. We describe the underlying ground-truth environment model in this section (which is not available to our agents) and move to agent's perspective (partial observability of states and limited set of action space) in Section IV-C and IV-D.

[2]When high-level decision-maker chooses *exploitation* as perception agent's next action, action agents receive the estimated state-information from perception agents and then start executing their manipulation task, which takes an upper-bound service time of $\mathcal{T}_{\mathcal{UB}}$ to be executed.

As introduced in Section III-B, the environment's state consists of targets' states $(s_t^{T_1}, \cdots, s_t^{T_{N_T}})$, perception agents' states $(s_t^{P_1}, \cdots, s_t^{P_{N_P}})$, and action (manipulator) agents' states $(s_t^{M_1}, \cdots, s_t^{M_{N_M}})$. Thus, the state of the environment is $s_t = (s_t^{T_1}, \cdots, s_t^{T_{N_T}}, s_t^{P_1}, \cdots, s_t^{P_{N_P}}, s_t^{M_1}, \cdots, s_t^{M_{N_M}}) \in \mathcal{S} = \{S_t^T\}^{N_T} \times \{S_t^P\}^{N_P} \times \{S_t^M\}^{N_M}$. Under the fully-observed state, the action space for each perception agent is straightforward: $a_t^{P_i} \in A_t^P = \{$exploit target 1, exploit target 2, $\cdots$, exploit target $N_T\}$. There is no need for exploration action as we already know the state information in the fully-observable case. In the example of wildfire, each exploitation action consists of three phases. First, a perception agent travels to an exploration target. Second, the perception agent extracts fire state information and generates the performance-guaranteed path for the action agent (Section V-B). Third, an action agent drops fire retardant following the received path. The target however, may not be fully fulfilled (e.g., the fire may not be completely extinguished) by a single manipulator's execution, and thus the state transition $T(s'|s,a)$ is stochastic. Therefore, perception agents may choose exploitation again to revisit the target. If the target has been fulfilled, the corresponding exploitation action will be removed. If the target has not been fulfilled, the perception agent generates a new trajectory, and calls in an action agent again. The state transition time $F(s,a,s')$ is dependent on the upper-bound service time, $\mathcal{T}_{\mathcal{UB}}$, for action agents that we derive in Section V-D. Unlike most SMDP formulations that have a reward rate during the entire transition time, in our setting, only an instant reward, r_t, is given on time t, when a target's task has been extinguished (e.g., extinguishing the fire at a target location). We argue that our instant reward setting is more semantically meaningful since longer execution on a target does not necessarily mean larger rewards. Upon completing the task for a target, the environment gives a reward proportional to the features of the target (e.g., a firespot's heat intensity). Thus, the total amount of reward is fixed when the initial state is given (the sum of all target importance). However, what we try to maximize is the discounted cumulative reward as per Eq. 3. The temporal discount factor γ will encourage faster completion of the tasks.

We further note that through introducing both 1) a multi-agent setting and 2) an SMDP over the typical MDP, we create a more complex and more broadly applicable problem to tackle. For example, consider the scenario in which Agent 1 is assigned to exploit Target 1 while Agent 2 is assigned to execute on Target 2. The execution time of each action, as mentioned in the previous paragraph, is a random variable dependent on $\mathcal{T}_{\mathcal{UB}}$. Therefore, the environment's state transition must consider the time ordering of agents' task executions. For the example above, if Agent 2 finishes first, the environment will transit to the state where Target 2 is processed and query the next action for Agent 2. Otherwise, the environment will, in turn, transit to the state where Target 1 is processed at first. We elaborate in more detail in Section IV-B. We emphasize that including just the multi-agent or SMDP facets would not create this challenge. Instead, it is through the combination of both (i.e., a multi-agent SMDP) that causes this complexity.

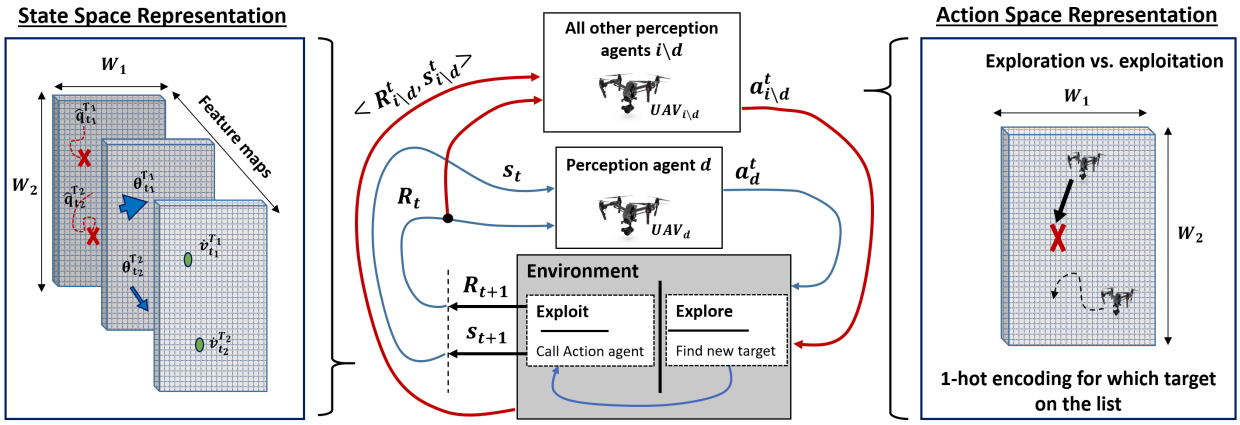


Figure 2: This figure depicts the designed, landscape-based, action and state space representation in the multi-agent Semi-MDP (middle). Our action-space (right side) can be represented as $\mathcal{A} \in \{0,1\}^{w_1 \times w_2}$, where an entry $\mathcal{A}_{i,j} = 1$ indicates the location of an *exploitation* action's target, and when $\mathcal{A}_{i,j} = 0, \forall i,j \in \mathbb{N}$ an *exploration* action is declared by the decision-making module. We take a similar approach to represent the state-space (left side) as overlaid feature maps forming a feature tensor $\mathcal{S} \in \{S_t\}^{w_1 \times w_2 \times k}$.

B. Discrete-Event, Continuous Time Environment Simulation

In the majority of prior decision-making work [35, 36, 37, 38, 39], time is discretized by constant intervals (fixed-increment time progression) à la an MDP. However, discrete time-steps are not suitable for modeling our environment. Since we are in a multi-agent asynchronous execution setting, it is possible that before the current agent's action finishes, another agent finishes its task and queries for a new task, as described in the example in Section IV-A. Therefore, we consider a discrete-event continuous-time progression for our environment that is more precise and efficient (see Fig. 3).

In our setting, we define an event is when an agent finishes its current task and becomes idle. At such event, the decision-making algorithm needs to assign a new task to the agent given the current state. As the environment has perfect information about the targets, the robots' motion restrictions (e.g., maximum velocity and turning bank), and the upper-bound service time ($\mathcal{T}_{\mathcal{UB}}$) (as derived in Section V-D), it could calculate the ground-truth time distribution for an assigned task. Accordingly, the simulation can readily determine the sequence of events, and thus, we can implement continuous time progress via discrete-event callback for the decision-making algorithm. In the discrete-event simulation, we can rewrite Eq. 3 as $R = \sum_{t_i \in \mathcal{T}} \gamma^{t_i} r_{t_i}$, in which $\mathcal{T}$ represents a set of times for all discrete-events. The event set, $\mathcal{T}$, is important in that it bridges the continuous time process $\{s_t, a_t, r_t | t \geq 0\}$ and discrete chain $\{s_{t_i}, a_{t_i}, r_{t_i} | i \in \mathbb{N}\}$.

C. Observation and Action Space Representations

The agents and the underlying high-level decision-making algorithm do not have perfect information regarding the state and instead receive observations, $o \in \Omega$, which contain information about known targets. The action space available to each agent is not the entire target list since we have no prior information about the number of targets or target locations. Thus, we need to have an exploration action that is to find new targets. Further, the set of potential targets we could exploit is dynamic as exploration will result in uncovering new targets and thus increasing the target list length. Even with the simplifying assumption such that, for a single agent, exploration adds at most one new target, a conventionally defined action-space (e.g., discrete action set of choices to explore or which target to exploit) will be variable-length, and a canonical neural network cannot deal with such a dynamic length action space. Moreover, the agents also need to revisit previously exploited targets, as described in Section IV-A.

Accordingly, we introduce a "landscape-based" action representation, $a \in \{0,1\}^{w_1 \times w_2}$ (illustrated in Fig. 2), a $w_1 \times w_2$ binary matrix in which w_1 and w_2 represent the environment x and y dimensions. For exploitation action, matrix $\mathcal{A}$ is a one-hot encoding, and the corresponding 1 location on the terrain is the target location for exploitation. We further define an all 0 matrix representing exploration action. In this work, perception agents follow a predefined horizontal sweeping exploration strategy, but we also note the possibility of learning an exploration policy under our proposed framework. With such representation, the action space $\mathcal{A}$ now has constant size $(1 + w_1 \times w_2)$, and semantic ambiguity will be avoided. We can dynamically generate a small subset of available actions each time with an observation. As shown in Fig. 2, we take a similar approach to represent the observation-space as overlaid feature-maps forming a feature tensor of form $o \in \mathbb{N}^{w_1 \times w_2 \times k}$ where k is the number of features. The Observation space includes all targets that have been discovered so far, whether or not they are currently inside the respective UAV's FOV.

D. Multi-Agent Partially Observable Semi Markov Decision Process (MA-POSMDP)

We define a novel problem setup termed Multi-Agent Partial Observable Semi Markov Decision Process (MA-POSMDP) as a 9-tuple $(\mathcal{S}, \Omega, O, \mathcal{A}, r, T, F, \gamma, \rho_0)$ [29]. State space $\mathcal{S}$ is introduced in Section IV-A, and state $s_t \in \mathcal{S}$ consists of all the current information of the world for the process $\{s_t | t \geq 0\}$. Observation space, Ω, is introduced in Section IV-C, and the current observation is given by the current state via observation model O: $o_t \sim O(\cdot | s_t)$. $\mathcal{A}$ is the action-space, and we leverage the landscape-based action representation described in Section IV-C. We reiterate that despite the huge size of the entire action space, the possible action space under each state is

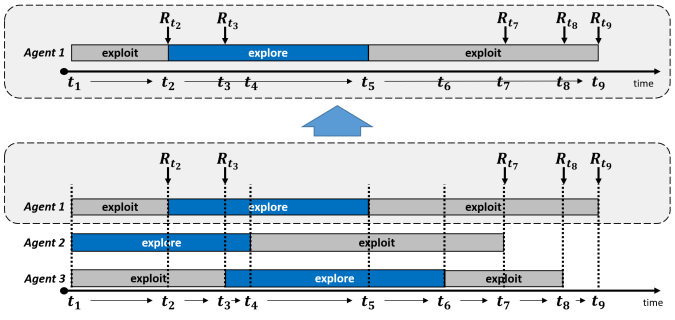


Figure 3: This figure depicts an example of timings and transitions in our discrete-event simulation of continuous time progression. The top gray panel represents the transitions in agent 1's perspective.

very limited, being $l+1$ with l known targets and one explore action. $\gamma \in [0, 1)$ is the temporal discount factor for each unit of time and $\rho_0(s)$ is the initial state distribution. $r(s, a)$ represents the reward when execution of action a on state s is finished, (i.e., an impulse function which has non-zero values only when an exploitation action is finished), and the amount of reward is relative to the priority/severity of the exploited target (e.g., fire intensity as in Section II-A in the aerial wildfire fighting example). As such, we do not give explicit reward for exploration and discovering new targets, as our objective is defined on the team's discounted cumulative rewards instead of on each agent's individual rewards. This overarching reward is designed to encourage prolific cooperation between perception and action agents, and the learning mechanism, which will be introduced in the next section, will further enhance the collaboration. $F(s, a, s')$ is the time required for executing action a in state s and transit to s', which is determined by the introduced upper-bound time for service $\mathcal{T}_{\mathcal{UB}}$ and robots' dynamics (e.g., maximum velocity) in our proposed low-level component (Section V-D). Transition $T(s'|s, a)$ encodes the effect of agent's action a on state s and is also dependent on the time ordering as shown in Section IV-A and IV-B. Note: we make the assumptions that agents do not fail and that the underlying motion model of the targets is known, and thus, the predicted duration time for actions are reliable.

Agents start with an initial belief over states, which, depending on the underlying application, could either encode a prior belief of some targets or be a non-informative prior. When a perception agent P_i is idle at time, t, its current observation information $o_t \sim O(\cdot|s_t)$ is fed into the decision algorithm to query for an action a_t. The action is executed in the environment which calculates the next state $s' \sim T(\cdot|s_t, a_t)$, the required time for the action $f_t \sim F(s_t, a_t, s')$, and a reward $r_t \sim r(s_t, a_t)$. Since we are in a multi-agent, asynchronous execution setting, it is possible that before the current agent's action finishes, another agent finishes its task and queries for a new task. Therefore, we utilize a discrete-event simulation as described in Section IV-B.

We rewrite our optimization goal in Section IV-B to introduce the optimization variable (policy π) in Eq. 4:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathcal{R}(\pi) = \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t_i \in \mathcal{T}} \gamma^{t_i} r_{t_i} \right] \quad (4)$$

Despite the similarity of the objective to standard MDP,

Algorithm 1: MA-POSMDP

1: Initialize Global_Step= 0, Q-network Q_θ and target Q-network Q^*_θ
2: **while** not converged **do**
3: Obtain from the low-level module the upper-bound service time $\mathcal{T}_{\mathcal{UB}}$, from one of the Eq. 23, Eq. 25 or Eq. 28 for current task and manipulator agent
4: Obtain rollout $\tau = \{\langle t_i, s_{t_i}, a_{t_i}, r_{t_i} \rangle\}$ in which time is determined via $\mathcal{T}_{\mathcal{UB}}$ in line (3)
5: Transform rollout τ to single-agent perspective transitions $\langle s_{t_i}, a_{t_i}, \Delta t_i, r_{t_j}, s_{t_j}, a_{t_j} \rangle$ via Eq. 8 and calculate $\mathcal{B}_i = \sum_{k=i+1}^{j} \gamma^{t_k - t_i} r_{t_k}$
6: Store $\langle s_{t_i}, a_{t_i}, \Delta t_i, \mathcal{B}_i, s_{t_j}, a_{t_j} \rangle$ to replay buffer
7: **for** i = 1 to iters **do**
8: Sample transitions $\langle s, a, \Delta t, \mathcal{B}, s', a' \rangle$ from replay buffer
9: Train Q_θ via Eq. 9
10: Global_Step $\leftarrow$ Global_Step $+1$
11: **if** mod (Global_Step, Update_Interval) $== 0$ **then**
12: Update target network, $\theta^* \leftarrow \theta$
13: **end if**
14: **end for**
15: **end while**
16: **return** Q_θ

we note that t_i in our MA-POSMDP setting in Eq. 4 is a continuous value from a finite set $\mathcal{T}$ containing all events' times. We note that, our high-level decision making is a centralized process among perception agents. Our proposed MA-POSMDP learning process and the overall interaction between high-level and low-level modules are summarized in the Algorithm 1.

E. Multi-Agent SARTSA Learning

We first assume that all perception robots are homogeneous and can thus share the same decision-making module, specifically in this work, a Neural Network (NN). We note that heterogeneity still exists between perception agents and action agents. We extend the SARSA learning algorithm [40] for our particular MA-POSMDP formulation.

SARSA learning is based on the 1-step Temporal Difference (TD) signal in a typical MDP, relying on the transition tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ as shown in Eq. 5.

$$\delta_t^{\text{1-step TD}} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \quad (5)$$

SARSA's learning rule is given by Eq. 6 in which α is the learning rate.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t \quad (6)$$

SARSA has also been extended to incorporate n-step TD error, which is a balance between 1-step TD estimation and

Monte-Carlo (MC) estimation for the entire trajectory, as shown below in Eq. 7.

$$\begin{aligned}&\delta_t^{\text{n-step TD}}\\&= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s_t, a_t)\\&= \left(\sum_{k=t+1}^{t+n-1} \gamma^{k-t-1} r_k\right) + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s_t, a_t)\end{aligned} \tag{7}$$

MC estimations update each state based on the entire sequence of observed rewards from the current state until the end of episode and 1-step TD learning performs the updates only based on the next-step reward. The N-step TD method, however, combines the two and applies updates based on observed rewards in next N steps. The N-step TD approach has shown its effectiveness and sample-efficiency in several well-known RL algorithms such as A3C [35].

In MA-POSMDP, we have to first take observations as if they are actual state, as we are limited to the information. We have two options to consider the SARSA-like transition tuples. First we can consider all the adjacent transitions, i.e. $\langle s_{t_i}, a_{t_i}, r_{t_{i+1}}, \Delta t_i, s_{t_{i+1}}, a_{t_{i+1}}\rangle$ in which $\Delta t_i = t_{i+1} - t_i$. However, the transition from s_{t_i} to $s_{t_{i+1}}$ does not necessarily come from a_{t_i}, as there might be other actions that finish before a_{t_i}. For example, in Fig. 3, Agent 2's second action will transit to the end of Agent 1's second action; however, the consequent reward is not from Agent 2's action. Therefore, such a transition tuple will create an incorrect signal for learning algorithms which can be problematic.

Second, we can view transitions on a single agent's timeline as shown in Eq. 8, where $j > i$ and, t_j and t_i are events for the same agent. Moreover, $j = \min_{k>i} k$, such that event k and event i are for the same agent and, $\Delta t_i = t_j - t_i$.

$$\langle s_{t_i}, a_{t_i}, \Delta t_i, r_{t_j}, s_{t_j}, a_{t_j}\rangle \tag{8}$$

For example, in the perspective of Agent 1, the transition of the second action is from t_2 to t_5. As such, the new state s_{t_j} (e.g. s_{t_5}) contains the influence of a_{t_i} (e.g., a_{t_2}). It could also contain other agents' action effects (e.g., a_{t_1} of Agent 2 and 3). However, we could view the process between t_i and t_j as executing the same policy π for $j-i$ times because all agents share the same policy. For instance, the transition between t_2 to t_5 could be viewed as applying the current policy three times, and we receive r_3, r_4, r_5 on t_3, t_4, t_5, respectively. Through such view, we obtain a novel TD signal (Eq. 9) similar to the $j-i$ step TD error (Eq. 7).

$$\delta_t^{\text{MA-TD}} = \left(\sum_{k=i+1}^{j} \gamma^{t_k - t_i} r_{t_k}\right) + \gamma^{\Delta t_i} Q(s_{t_j}, a_{t_j}) - Q(s_{t_i}, a_{t_i}) \tag{9}$$

For implementation, we propose to maintain a reward buffer $\mathcal{B}_i$ for each agent i to account for any reward the team has received collectively during the period when agent i is executing an action. For instance, $\mathcal{B}_1$ will record r_3, r_4 on t_3, t_4 during t_2 to t_5.

Thus, we could utilize $\delta_t^{\text{MA-TD}}$ to achieve SARSA-like learning by Eq. 6 and all transitions in the view of each agent. We term this learning algorithm as Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) as it explicitly considers time and multi-agent setup.

V. Low-level Coordinated Control and Planning

The lower-level in our hierarchical algorithm structure is a distributed, coordinated control and planning framework through which robots in the composite team will execute tasks (i.e., exploration or exploitation as assigned by the high-level decision-maker). The low-level control module is responsible for sequencing the tasks while accounting for robot's specific traits (capabilities) and motion restrictions. To this end, we first solve a coordinated routing problem between perception and action robots (Section V-B). Action robots are not capable of sensing the dynamic targets and, therefore, need to receive predicted future target states. Thus, to maximize target track quality, a probabilistic framework is developed in Section V-C to propagate and evaluate measurement error residuals over time. We leverage the derived tracking error upper-bound (TEB) to compute the upper-bound time for action agents' service, $\mathcal{T}_{\mathcal{UB}}$, in Section V-D which is also used for the task timings in our MA-POSMDP formulation underlying the high-level decision-maker (Section IV-D). $\mathcal{T}_{\mathcal{UB}}$ is the upper-bound time it takes for an action agent to finish its task (e.g., travel to fire locations and extinguish fire) after receiving the required state-information from the perception agent.

The rest of Section V is organized as follows: in Section V-A, we tackle the problem of online target state estimation via UAV sensors (given the action from the high-level decision-maker decides exploitation) through EKF estimation. We investigate the time-dependency of EKF's measurement uncertainty in Section V-A2 as a prerequisite for validating our covariance-based planning. Then, in Section V-B, we propose our low-level scanning framework to improve resiliency and cooperation efficiency between perception and action agents. Eventually, we propose our analytical tracking error upper-bound in Section V-C and derive a set of probabilistic upper-bound service times for action agents in Section V-D.

A. Preliminaries: Target State Estimation and Measurement Uncertainty Propagation via UAV Perception Agents

Due to their agility and hovering capability, quadcopter UAVs are good candidates for real-time dynamic target tracking. The problem of state estimation and measurement error propagation has been widely studied in the classical state estimation literature [41, 42, 43]. Although there are more sophisticated and complex approaches available for multi-target tracking (see [44]), we utilize Extended Kalman Filter (EKF) as a suitable tool to both estimate the target states and to propagate the measurement uncertainty residuals due to the model and sensor inaccuracies [45, 41]. EKF locally linearizes the target's nonlinear motion model, $\mathcal{M}_t$ (e.g., the fire propagation model in Eq. 1), and perception agent's observation model, $\mathcal{O}_t$ (Fig. 4), about the estimate of the current mean and covariance, and thus $\mathcal{M}_t$ and $\mathcal{O}_t$ do not need to be linear functions of the state, but may rather be differentiable functions.

Considering a dynamic target on the ground, $q_t = [q_t^x, q_t^y]$, moving according to a nonlinear model, $\mathcal{M}_t$ (e.g., Eq. 1), at each time, t, the observation mapping, $\mathcal{O}_t$, of a flying

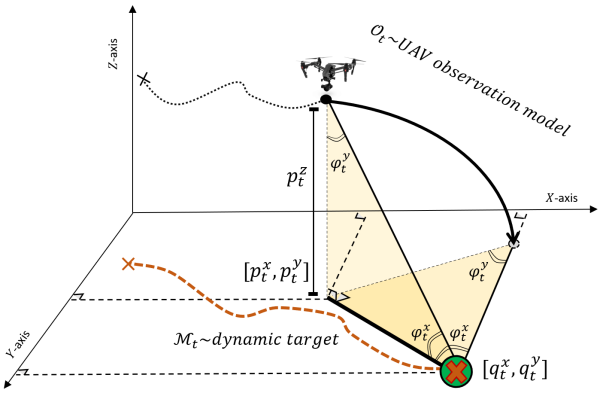


Figure 4: This figure depicts a flying UAV's (perception agent) observation model with respect to a dynamic target on the ground (the target altitude $q_t^z = 0$). The angle parameter, φ_t, represents the angle by which the UAV is observing the target on the ground.

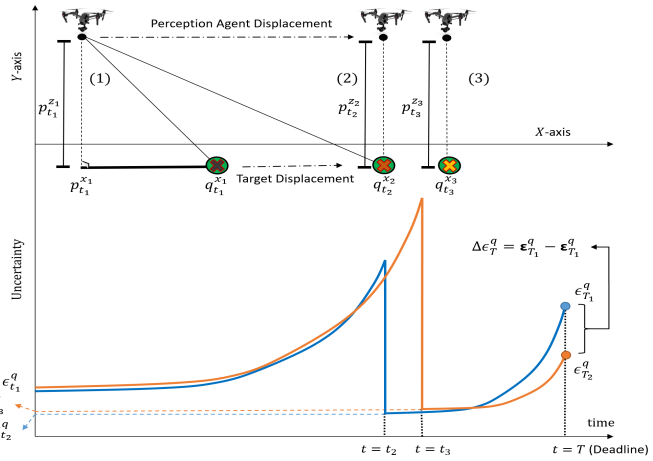


Figure 5: This figure depicts the time-dependency of measurement uncertainty as propagated by EKF. The dropping uncertainty regardless of the time of visit will always have the same value ($\epsilon_{t_2}^q = \epsilon_{t_3}^q$), if the observing robot's displacement to target is similar at both times.

perception agent with pose, $p_t = [p_t^x, p_t^y]$ and altitude p_t^z, with respect to the target's location can be shown as in Eq. 10.

$$\mathcal{O}_t : \{q_t, p_t\} \rightarrow \varphi_t : \|q_t - p_t\|_2 = \|q_t^z - p_t^z\|_2 \tan\varphi_t \quad (10)$$

Given $\mathcal{S}_{t-1}$ as the current joint state vector of the dynamic target and perception agent's states; desired is an estimated state vector, $\hat{\mathcal{S}}_t$, one step forward in time, given the current target position distribution, $q_{t|t-1}$, target's motion model with current parameters ($\mathcal{M}_{t|t-1}$), and perception agent's observation model of the target ($\mathcal{O}_{t|t-1}$). In other words, we seek to estimate the following joint Probability Density Function (PDF), ρ, in Eq. 11 through the EKF.

$$\hat{\mathcal{S}}_t = \arg\max_{\mathcal{S}_t} \rho\left(q_{t|t-1}, p_{t|t-1}, \mathcal{M}_{t|t-1}, \mathcal{O}_{t|t-1}\right) \quad (11)$$

In the application of aerial wildfire fighting where perception agents are tasked to seek out the state information of propagating firespots, we formulate this estimation with the EKF's state transition and observation equations as in Eq. 12 and Eq. 13, respectively.

$$\left[\hat{\mathcal{S}}_t\right]_{8\times1} = \left[\left.\frac{\partial\mathcal{M}_t}{\partial\mathcal{S}_i}\right|_{\hat{\mathcal{S}}_{t|t-1}}\right]_{8\times8}\left[\mathcal{S}_{t-1}\right]_{8\times1} + \omega_t \quad (12)$$

$$\left[\hat{\Phi}_t\right]_{5\times1} = \left[\left.\frac{\partial\mathcal{O}_t}{\partial\Phi_i}\right|_{\hat{\Phi}_{t|t}}\right]_{5\times8}\left[\hat{\mathcal{S}}_t\right]_{8\times1} + \nu_t \quad (13)$$

In these equations, $\mathcal{S}_t = [q_t^x, q_t^y, p_t^x, p_t^y, p_t^z, R_t, U_t, \theta_t]^T$ is the joint state vector, ω_t and ν_t are the process and observation noises, which are modeled by zero mean white Gaussian random variables, respectively, to account for stochasticity in fire behavior and inaccuracies in fire propagation and perception agents' observation models. $F_t = \partial\mathcal{M}_t/\partial\mathcal{S}_i$ is the state transition Jacobian matrix in which the FARSITE model introduced in Section II-A is used as the transition model to derive the partial derivatives with respect to all of the state variables in $\mathcal{S}_t$. The observation Jacobian matrix, $H_t = \partial\mathcal{O}_t/\partial\Phi_i$, is a mapping model, as shown in Fig. 4, through which the predicted fire propagation model parameters and UAV locations are translated into a unified angle-parameter vector $\hat{\Phi}_t = \left[\varphi_t^x, \varphi_t^y, \hat{R}_t, \hat{U}_t, \hat{\theta}_t\right]^T$. We note that, although the angle-parameters, φ_t^y and φ_t^x, are complementary angles, we utilize both angles as our goal is to use these angles for tracking 2D pose uncertainty (and not just angle uncertainties). A detailed derivation of EKF equations, Jacobian matrices, and uncertainty propagation is provided in the Appendix-A.

*1) **EKF Bayesian Posterior and Minimum Mean-Squared Error Estimate**:* EKF is an approximate Bayesian filter, and taking the resulting Gaussian distribution as the true Bayesian posterior can be imperfect in some localization and tracking applications. Nevertheless, we note that, in this work, we do not require the true Bayesian posterior, and none of our planning/decision-making algorithms are based on a true Bayesian posterior. Assuming a given map, $\mathcal{S}$ (e.g., a set of discrete world features or states), and a sequence of target relative observations, $\mathcal{Z}$, described by the conditional probability, $p(\mathcal{Z}|\mathcal{S}, q)$, we seek to estimate the PDF of the variable q, as $p(q|\mathcal{S}, \mathcal{Z})$ through EKF. For this problem, it can be shown mathematically [46, 47] that if we parameterize the random vectors q and $\mathcal{S}$ with mean and variance, then the EKF will compute the Minimum Mean-Squared Error (MMSE) estimate of the posterior. In our coordination framework, this MMSE estimate is an acceptable metric, in keeping with its use in prior localization and dynamic target tracking literature [48, 49, 50]. We leave improvements in state estimation and accurate target tracking to future work as they are not the focus of our current study.

*2) **Time-dependency of EKF's Measurement Uncertainty**:* Our analytical upper-bound time for service in Section V-D depends on the state-estimation measurement uncertainty; thus, we examine the time-dependency of the propagated error through the EKF. Considering the two examples presented in Fig. 5 in which a perception agent starts at position (1) at time $t = t_1$ with some distance from a target and visits the moving target either at $t = t_2$ or $t = t_3$ (i.e. with $\Delta t = t_3 - t_2$ latency in latter case), we show that the dropping measurement uncertainty of the target's state, regardless of the time of visit, is only a function of the distance between the perception agent and the target. To this end, we first define the uncertainty drop in our scenario in Lemma 1; in Theorem 2, we prove that such

uncertainty drop is independent of time.

Lemma 1. *If the uncertainty (Kalman measurement residual) of a sensing robot observing a dynamic point q_t directly from distances $\Delta\mathcal{X}_{t_1}$ and $\Delta\mathcal{X}_{t_2}$ at times t_1 and t_2 (where $t_2 > t_1$) are defined by $\mathcal{E}^q_{t_1}$ and $\mathcal{E}^q_{t_2}$, respectively, then $\mathcal{E}^q_{t_2} < \mathcal{E}^q_{t_1}$ if and only if $\Delta\mathcal{X}_{t_2} < \Delta\mathcal{X}_{t_1}$. We define the uncertainty drop as follows in Eq. 14.*

$$\Delta\epsilon^q_{t_2,t_1} = \mathcal{E}^q_{t_2} - \mathcal{E}^q_{t_1} \tag{14}$$

Proof. It has been shown previously (see [51, 52, 53]) that a locally optimal strategy for state estimation is obtained by driving the robot to positions that maximize the prediction variance of the observation [53]. As such, minimizing the predicted state covariance (Eq. 15) corresponds to maximizing the covariance residual Λ_t in Eq. 16. According to Eq. 16, by setting the state covariance to identity ($\Sigma_{t|t-1} = I$) and keeping the noise covariance constant ($\Gamma_t = \Gamma$), we see that a maximally informative position for a robot is the one that minimizes $H_t H_t^T$. In other words, all other settings being equal, the closest possible position where dynamic observations change rapidly as a function of robot position [53] minimizes the determinant of the observation covariance and thus, minimizes the total uncertainty. Note that Λ_t is a function of both the state estimate and the map covariance $\Sigma_{t|t-1}$. ■

Next, we present Theorem 2 and a proof sketch. Please see Appendix-A for a detailed proof of Theorem 2.

Theorem 2. *Measurement uncertainty drop about the states of a dynamic point q_t as defined in Lemma 1 and Eq. 14, observed by a perception robot directly from a distance $\Delta\mathcal{X}$ at time T (i.e. $\mathcal{E}^q_T$), is independent of time. It is only a function of displacement $\Delta\mathcal{X}$ between the observer and the point.*

Proof. The model and observation measurement uncertainties associated with EKF estimation follow the general nonlinear uncertainty propagation law, shown in Eq. 15-16.

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + Q_t \tag{15}$$

$$\Lambda_{t|t} = H_t \Sigma_{t|t-1} H_t^T + \Gamma_t \tag{16}$$

where $\Sigma_{t|t-1}$ is the predicted covariance estimate, $\Lambda_{t|t}$ is the innovation (or residual) covariance, F_t and H_t are the process and observation Jacobian matrices, and Q_t and Γ_t are the process and observation noise covariances, respectively. Considering Eq. 15-16, changes in the uncertainty values occur through changes in the gradients in the process and observation Jacobian matrices (F_t and H_t). The gradients in the Jacobian matrices are calculated as derivatives of the target's motion model, $\mathcal{M}_t$ (Eq. 1), and the perception agent's observation model, $\mathcal{O}_t$ (Eq. 10), as in Eq. 17.

$$F_t = \left[\left. \frac{\partial \mathcal{M}_t}{\partial \mathcal{S}_i} \right|_{\hat{\mathcal{S}}_{t|t-1}} \right] \quad \text{and} \quad H_t = \left[\left. \frac{\partial \mathcal{O}_t}{\partial \Phi_i} \right|_{\hat{\Phi}_{t|t}} \right] \tag{17}$$

where $\mathcal{S}$ and Φ are the process and observation state-vectors, respectively. Accordingly, matrix F_t is time-invariant if the gradients in this matrix are time-invariant, which depends on the target's motion model. Considering position q and velocity $\dot{q}$ of a moving target as the state variables, it can readily be seen that the gradients of a numerical motion model such as $q_t = q_{t-1} = \dot{q}_{t-1}\delta t$ (similar to Eq. 1) with respect to its state variables are $\partial q_t / \partial q_{t-1} = 1$ and $\partial q_t / \partial \dot{q}_{t-1} = \delta t$ and are time-invariant for all constant time steps δt. Moreover, considering the observation model presented in Eq. 10 and positions and velocities of the sensing UAV and the target as the state variables in Φ, the gradients in H_t are only functions of the Euclidean distance between the perception agent and the target locations (see Appendix -A). Accordingly, both F_t and H_t are time-invariant and with constant process and observation noise covariances (Q_t and Γ_t), the total uncertainty drop is also not a function of time. See Appendix-A for a detailed derivation of the Jacobians and a rigorous proof. ■

B. Low-level Control Framework for Scanning

Perception agents utilize EKF to estimate a probability distribution and a measurement covariance for each state variable [54]. The first step in our coordinated routing framework for the perception-action composite robot team is for the perception agents to search the environment (e.g., move around and scan) and add newly found targets to a list. This list will be sent to an assigned action agent as its task. Such process of assigning tasks to action agents is generally known as a Multi-Robot Task Allocation (MRTA) problem [4]. We approximately solve this MRTA problem by tackling a constraint satisfaction problem with action agents as variables, tasks as domains, and constraints such as relative distance to the task locations, battery availability, etc. In our greedy solution, action agents are assigned to tasks based on their availability and their relative distance to the task location, since distance based assignments are fast and deployable in real-time.

Based on the underlying application, we leverage the Close-Enough Traveling Salesman Problem (CE-TSP) with Steiner zone variable neighborhood search [55] so that the action agent only needs to get "close enough" to each goal, according to a predefined $\Delta-$disk proximity. In the application of aerial wildfire fighting, the perception agent observes multiple fire-spots within its field-of-view (FOV); however, it does not need to pass all of these "nodes". Instead, the perception agent first identifies the overlapping $\Delta-$disks between k fire points as Steiner zones and then chooses the centroid nodes as coordinates to be added to the target list $\mathcal{L}_t$ [24].

While scanning the environment, perception agent takes into account the specific motion/flight model of the assigned manipulator robot. This process includes accounting for the action agent's maximum turning rate ω^M_{max} and minimum and maximum linear velocities $\left[v^M_{min}, v^M_{max}\right]$. The perception agent leverages this information alongside the state-information of the first sensed target, which is propagated $\mathcal{T}_{\mathcal{UB}}$ steps into the future according to the target's motion model $q_{t+\mathcal{T}_{\mathcal{UB}}}$, and explores for new targets only within reachable areas by the action agent. Note that $\mathcal{T}_{\mathcal{UB}}$ is the upper-bound time it takes for action agent M to travel to the target location after receiving the required state-information from the perception agent. This process is elaborated in Fig. 6 in which the blue areas represent the reachable polygons, the four vertex coordinates of which

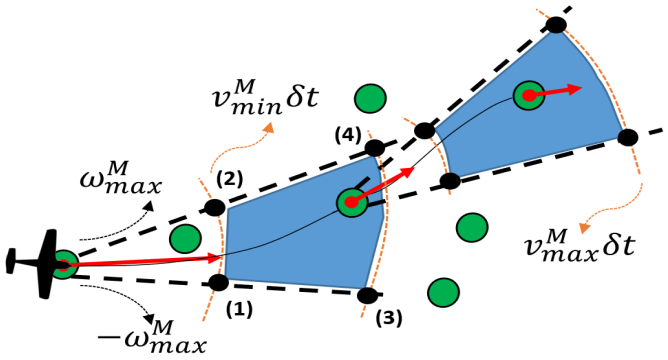


Figure 6: This figure depicts our coordinated routing problem. Perception agent accounts for the assigned manipulator agent's motion model, including its maximum turning rate ω^M_{max} (dashed black lines) and minimum and maximum linear velocities $[v^M_{min}, v^M_{max}]$ (red dashed lines). Sensing agent only scans the reachable areas for targets (green dots), enclosed by four vertices (1)-(4), (black dots).

$\vec{p}_i$ s.t. $i = 1, \cdots, 4$ can be calculated through $\vec{p}_i = \vec{p}_M + \vec{u}_i$, in which $\vec{p}_M = q_{t+\mathcal{T}_{\mathcal{UB}}}$ is the manipulator agent's position when it arrives at the propagated coordinates. Moreover, $\vec{u}_i$ represents the respective rotation vectors, rotating point $\vec{p}_M$ to $\vec{p}_i$ according to the manipulator agent's linear and angular velocities, which can be calculated according to Eq. 18.

$$\vec{u}_i = \mathcal{D}\delta t \left(\frac{\vec{v}}{\|v\|}\right) R^{\varphi}_z \left((-1)^i \varphi\right) \tag{18}$$

In Eq. 18, φ is the rotation-angle of the velocity-vector for one time-step, δt, due to maximum angular velocity and can be calculated as $\varphi = \omega_{max}\delta t/2$. $\mathcal{D}$ is the robot's planar displacement for one unit of time and equals $\mathcal{D} = v^M_{min}\delta t$ for $i = 1, 2$ and is $\mathcal{D} = v^M_{max}\delta t$ for $i = 3, 4$. Additionally, $R^{\varphi}_z(\varphi)$ is the rotation matrix around the z-axis. For a more detailed discussion of the above derivations, refer to Appendix-C.

Leveraging the proposed scanning procedure results in an action agent-friendly path where a manipulator robot with motion restrictions (e.g., a fixed-wing aircraft) can directly visit all of the determined waypoints with no limitations. As stated in Section IV-C, in our framework, perception agents can either follow any off-the-shelf exploration approach, such as sweeping or spiral paths, or they can learn to explore the environment through a modified action-space (to include primitive motion actions) and reward structure to learn an exploration policy. In Section VI, we evaluate sweeping and spiral paths as perception agents' exploration strategies. Moreover, perception agent's high-level action (e.g., explore the environment or exploit the list) is determined by the high-level MA-SARTSA algorithm and thus, while action agents are servicing the task, the perception agent could be exploring the environment or exploiting targets for another action agent.

C. Analytical Tracking-Error Bound

When exploiting a target, the perception agent estimates targets' states (e.g., position and velocity) to infer targets' motion dynamics and calculate two quantities: (1) the upper-bound time, $\mathcal{T}_{\mathcal{UB}}$, required for an assigned manipulator robot to reach the sensed node, $\hat{q}$, and (2) a prediction of measurement residual covariance, $\mathcal{T}_{\mathcal{UB}}$ steps into the future by repeatedly applying EKF's prediction and update steps $\mathcal{T}_{\mathcal{UB}}$ times (Eq. 12-13). This process obtains an MMSE estimate of the Bayesian posterior (see V-A1). Moreover, $\mathcal{T}_{\mathcal{UB}}$ is the upper-bound time it takes for an action agent to finish its task after receiving the required state-information from perception agent.

Note: our tracking-error bound is inspired by the uncertainty residual ratio introduced in prior work [24]; however, our uncertainty-based temporal upper-bound derivations here are derived with a different goal in mind for a different underlying problem. In [24], we introduced an analytical uncertainty-based bound that compares the time it would take for a firespot to escape a perception agent's FOV relative to the total time it would take for that perception agent to complete a tour of observations on a set of distant nodes. In this paper, we uniquely consider an analytical tracking-error bound that compares the time taken for a dynamic target's total uncertainty (e.g., error in target's state estimation) to grow to a predefined value, $\mathcal{E}$, versus the time it takes for the action agent to reach to the location of that target. In our application, this pre-defined value represents the maximum uncertainty over the target's state that would still allow for the action agent to effectively douse the target with retardant. Accordingly, decisions made by both the perception agents and the action agents are directly impacted by this bound. To derive our bound, we leverage the model and observation measurement uncertainties associated with EKF estimation, as shown in Eq. 19-20.

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + Q_t \tag{19}$$

$$\Lambda_{t|t} = H_t \Sigma_{t|t-1} H_t^T + \Gamma_t \tag{20}$$

where $\Sigma_{t|t-1}$ is the predicted covariance estimate, $\Lambda_{t|t}$ is the innovation (or residual) covariance, F_t and H_t are the process and observation Jacobian matrices, and Q_t and Γ_t are the process and observation noise covariances, respectively. We note that this multi-step uncertainty propagation process only holds for Linear Time-Invariant (LTI) systems. In our framework, we utilize EKF, which locally linearizes the nonlinear motion model. Moreover, according to Theorem 2 (see V-A2), the measurement uncertainty as propagated by EKF is time-invariant. Now, we introduce an analytical Tracking-Error Bound (TEB) in Eq. 21, in which t_0 is the current time, $\hat{s}^l_t \in \mathcal{L}_t$ is the l-th waypoint set of detected target coordinates shared among connected perception agents, q is the current node in $\hat{s}^l_t$, $\mathcal{E}$ is an acceptable error threshold and $\mathrm{Tr}(.)$ is the trace operation to sum the uncertainties of all state-variables in the residual covariance matrix.

$$\mathrm{TEB}^q_{\mathcal{T}_{\mathcal{UB}}} = \frac{\mathrm{Tr}\left(\Lambda_{t_0+\mathcal{T}^q_{\mathcal{UB}}|t}\right)}{\mathcal{E}} \leq 1, \forall q \in \{\hat{s}^l_t\} \tag{21}$$

$\mathrm{TEB}^q_{\mathcal{T}_{\mathcal{UB}}}$ is an indicator of the scale to which the action agent might miss the sensed dynamic target. A TEB greater than one demonstrates a quickly/stochastically moving target for which the chance of being missed by the action agent is high and vice versa. We note that selection of the parameter $\mathcal{E}$ is application-dependent, where it can be tuned to a small value for highly sensitive/accurate tasks and vice versa. The perception agent

will use the TEB to determine the length, c, of the waypoint set, $\hat{s}_t^l = \{\hat{q}^{m_1}_{t+\mathcal{T}_{\mathcal{UB}}}, \cdots, \hat{q}^{m_c}_{t+\mathcal{T}_{\mathcal{UB}}}\}$, sent to the action agent m.

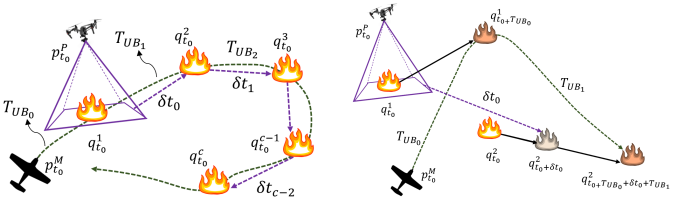
D. Probabilistic Upper-Bound Time for Service

In this section, we derive an analytical probabilistic upper-bound time, $\mathcal{T}_{\mathcal{UB}}$, which determines the upper-bound time required for the assigned action agent to reach to a "close-enough" proximity to the determined coordinates and provide service (i.e., extinguishing a fire in aerial wildfire-fighting). $\mathcal{T}_{\mathcal{UB}}$ is directly modelling the state transition times, $F(s, a, s')$ (i.e., task duration), in our MA-POSMDP environment, as described in Section IV-D. The number of waypoints in a set is chosen by determining the largest number of waypoints that can be included such that TEB (Eq. 21) is satisfied at each timepoint for all targets captured by the waypoints. If the TEB bound does not hold for a target, the perception agent stops adding targets (e.g., target states) to the set and passes the generated trajectory to the action agent. The target's motion model is a key element in this upper-bound time (e.g., moving pattern). Accordingly, we derive three bounds, one for each of the following scenarios: (1) stationary targets, (2) dynamic targets, and (3) moving and spreading targets. These service upper-bound times are used both for calculating the TEB in Eq. 21 in each scenario as well as in our high-level decision-maker as described in Section IV-D.

Following a similar reasoning as in [24], for the derivations, we reason about the velocity of the target (e.g., fire) at the α confidence level[3]. Further, in accordance with [24], we make two simplifying worst-case assumptions that (1) each target's velocity is conditionally independent given latent target dynamics (explicitly estimated in our EKF model) and (2) all targets uniformly move at their fastest velocity in the x- and y-directions. As such, the probability that our service time upper-bounds for action agents are accurate is demonstrated in Eq. 22, where $\mathcal{T}^*$ is the actual maximum time it takes for an action agent to get to the target's location from its current position, without exceeding the minimum track quality, $\mathcal{E}$, introduced in Eq. 21, with $\mathcal{T}_{\mathcal{UB}}$ as our bound. The probability that our bounds are *correct* means the probability that our derived temporal upper-bounds are in fact lower-bounded by this actual maximum time ($\mathcal{T}^*$) [24].

$$Pr\left[\mathcal{T}_{\mathcal{UB}} \geq \mathcal{T}^*\right] \leq 1 - (1-\alpha)^{\left|\hat{s}_t^l\right|} \tag{22}$$

1) ***Case 1: Stationary Targets****:* The first scenario we consider involves multiple stationary targets. Fig. 7a represents the coordination scheme for this case for two different types of UAVs as perception and action agents. Considering the time required for perception agent, P, to traverse between points n and $n+1$ represented as δt_{t_n}, we note that for a target with location q_{t_0} at time t_0, we have $q^n_{t_0} = q^n_{t_0+\delta_{n-2}}, \forall n \in \{\hat{s}_t^l\}^c$ (c is the length of l-th waypoint set) since targets do not move significantly in this case. Accordingly, the service upper-bound time for action agent, M, in Fig. 7a to visit all c target locations in the l-th waypoint set can be calculated according to Eq. 23. $p^M_{t_n}$ in Eq. 23 is the n-th position of the action agent before travelling to the $(n+1)$-th target position. We note that, $p^M_{t_0}$, is known as action agent's current position and for the following targets we have $p^M_{t_n} = q^n_{t_n}$, that is, the action agent will move towards $(n+1)$-th target starting from the position of the n-th target, $q^n_{t_0}$. In Case 1, we have $q^n_{t_n} = q^n_{t_0}$.



(a) Case 1: Stationary Targets (b) Case 2: Dynamic Targets

Figure 7: This figure depicts collaborative planning for coordinating between perception and action agents (UAVs in this case). Fig. 7a and 7b represent the stationary and dynamic target scenarios, respectively. Note that the moving targets in the second case (black arrows) in Fig. 7b, lead to a delayed exploitation by the perception UAV.

$$\mathcal{T}^{(C1)}_{\mathcal{UB}} = \sum_{n=0}^{c-1} \left(\frac{\left\| q^{n+1}_{t_0} - p^M_{t_n} \right\|}{\|v^M_{max}\|} \right) \tag{23}$$

Now, $\mathcal{T}^{(C1)}_{\mathcal{UB}}$ will be checked by the perception agent in the TEB bound in Eq. 21 to hold for each node at all times. A list of length c will be passed to the action agent if TEB is not satisfied for point $c+1$.

2) ***Case 2: Dynamic Targets****:* The second scenario includes dynamic targets with a known motion model (e.g., Eq. 1). In the case presented in Fig. 7b, targets move, which leads to a delayed exploitation with δt_{t_n}. Consequently, the service upper-bound time for an action agent to get to the point c can be calculated according to Eq. 24.

$$\mathcal{T}^{(C2)}_{\mathcal{UB}} = \mathcal{T}_{\mathcal{UB}_0} + \delta t_0 + \cdots + \mathcal{T}_{\mathcal{UB}_{c-1}} + \delta t_{c-2} \tag{24}$$

$$= \sum_{n=0}^{c-1} \left(\frac{\left\| q^{n+1}_{t_0+\mathcal{T}^{(C2)}_{\mathcal{UB}_{n-1}}} - p^M_{t_n} \right\|}{\|v^M_{max}\|} \right) \tag{25}$$

We note that, $p^M_{t_n}$ in Eq. 25 has a similar reasoning as in Eq. 23. In Eq. 24, δt_{c-2} is the traverse time it takes perception agent, P, with position $p^P_{t_0}$ and maximum velocity v^P_{max} to get from point $q^{c-1}_{t_0}$ to point $q^c_{t_0}$ and can be computed as in Eq. 26.

$$\delta t_{c-2} = \frac{\left\| q^c_{t_0+\mathcal{T}^{(C2)}_{\mathcal{UB}_{n-1}}} - p^P_{t_0+\mathcal{T}^{(C2)}_{\mathcal{UB}_n}} \right\|}{\|v^P_{max}\|}, \forall c \geq 2 \in \{\hat{s}_t^l\}^c \tag{26}$$

Note that $p^P_{t_0} = q^{c-1}_{t_0}$. Moreover, $q^{n+1}_{t_0+\mathcal{T}^{(C2)}_{\mathcal{UB}_{n-1}}}$ is the next target location to visit and can be obtained as shown in Eq. 27.

$$q^{n+1}_{t_0+\mathcal{T}^{(C2)}_{\mathcal{UB}_{n-1}}} = \frac{q^{n+1}_t \left(v^M_{max}\right)^2 - p^M_{t_n} \left(\dot{q}^{n+1}_t\right)^2}{\left(v^M_{max}\right)^2 - \left(\dot{q}^{n+1}_t\right)^2} \tag{27}$$

Similar to Case 1, $\mathcal{T}^{(C2)}_{\mathcal{UB}}$ is checked in TEB bound in Eq. 21 to hold for each node at all times. The list will be cut at node c if TEB does not hold for point $c+1$.

[3] $\alpha - \zeta^\alpha$ means the probability of the target being quicker than ζ^α is α. In other words $Pr[\zeta < \zeta^\alpha] = 1 - \alpha$ in which $\zeta^\alpha = \arg\max_{q,q'} \sqrt{\left(\widehat{\dot{q}^x_t}\Big|_\alpha\right)^2 + \left(\widehat{\dot{q}^y_t}\Big|_\alpha\right)^2}$

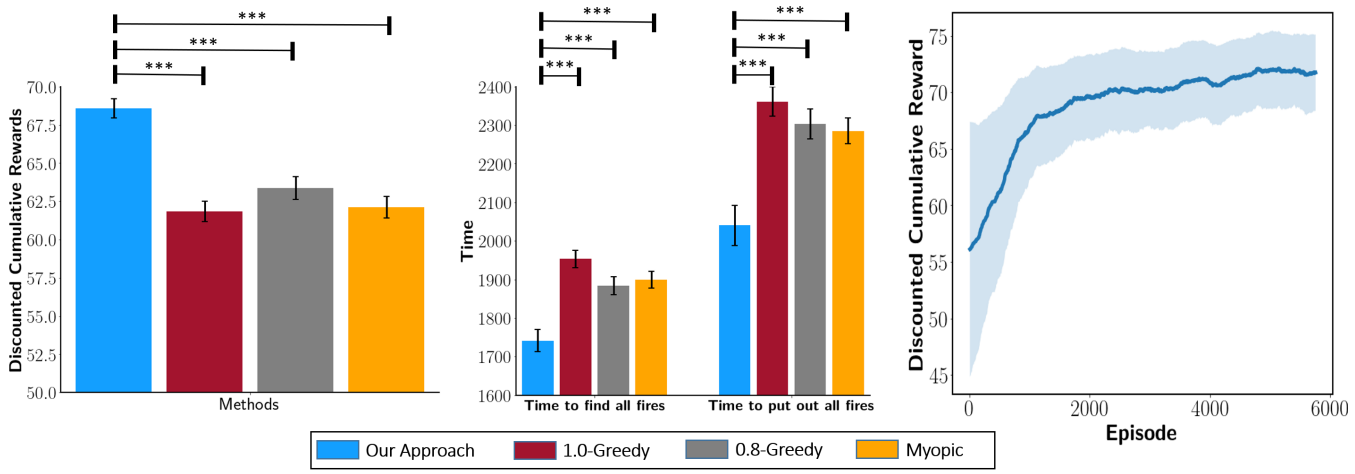


Figure 8: This figure depicts our benchmark comparison results. The leftmost figure shows a comparison on reward between our approach and benchmarks. Test repeated for 100 times. Error bars are standard error. The middle chart depicts a comparison on time (unit time in simulation) taken to find all fires and put out all fires. Test repeated for 100 times. Error bars are standard error. The rightmost figure shows our RL agent's training reward over the course of training averaged over three runs, where the shaded region depicts the standard deviation.

*3) **Case 3: Moving-Spreading Targets**:* The third scenario involves the case of targets that move and spread. This case is of particular interest in applications such as aerial wildfire fighting and oil-spill control in oceans via composite teams of robots. Here, single nodes of targets expand over time and thus can escape the perception agent's FOV. Assuming the spreading target's centroid as a dynamic node, the service upper-bound time, $\mathcal{T}_{\mathcal{UB}}$, for the action agent in Case 3 can be derived through a procedure similar to Case 2 and thus $\mathcal{T}_{\mathcal{UB}}^{(C3)} = \mathcal{T}_{\mathcal{UB}}^{(C2)}$ holds. Nevertheless, unlike the other two cases, $\mathcal{T}_{\mathcal{UB}}^{(C3)}$ cannot be used directly in the TEB bound in Eq. 21 to generate the waypoint set. We note that as targets grow out of the perception agent's FOV, a new Steiner area in our CE-TSP framework will be created and thus, a new separate waypoint will form, leading to losing track for the centroid location of the current target. Accordingly, an additional condition monitoring the target's spread needs to be checked in addition to TEB to determine the length c of waypoint set in Case 3, as defined in Eq. 28.

$$\mathcal{T}_{\mathcal{UB}_c}^{(C3)} \leq \mathcal{T}_{max}^{c}, \forall c \in \{\hat{s}_t^l\}^c \tag{28}$$

Eq. 28 is designed to check if the time it takes a spreading target, c, to escape the perception agent's FOV, ($\mathcal{T}_{max}^{c}$), is greater or smaller than the upper-bound service time, $\mathcal{T}_{\mathcal{UB}_c}^{(C3)}$, of the action agent. As such, while the above condition holds, the process continues as before and a set of size c will be passed to action agent if TEB is not satisfied for point $c+1$. The escaping time $\mathcal{T}_{max}^{c}$ can be estimated as shown in Eq. 29, following the outline of [24].

$$\mathcal{T}_{max}^{c} = \frac{-\beta + \sqrt{\beta^2 - 4\mu\delta}}{2\mu}, \forall c \in \{\hat{s}_t^l\}^c \tag{29}$$

The parameters μ, β and δ in Eq. 29 can be calculated as $\mu = \frac{4|\hat{s}_t^l|_t \dot{q}_t^2}{\mathcal{W}_t v_{max}^P}$, $\beta = 1 + \frac{2|\hat{s}_t^l|_t \dot{q}_t}{v_{max}^P}$ and $\delta = \frac{2\Delta L_t}{v_{max}^P\left(1-2\dot{q}_t\left(|\hat{s}_t^l|_t - 1\right)\right)}$, in which $|\hat{s}_t^l|_t$ is the length of the waypoint set at time t, $\mathcal{W}_t$ is the current width of the perception agent's FOV, $\dot{q}_t$ is the target's current total velocity in *XY*-coordinates and $\Delta L_t = \sum_{n=0}^{c-1} \left\| q_t^{n+1} - q_t^n \right\|$ is the total travel distance for the action agent between the last node added to the l-th waypoint set $\hat{s}_t^l$ and the first node. A detailed derivation for Eq. 29 can be found in Appendix-B.

Finally, to account for action UAVs' battery constraints, $T_t^{B_M}$, we directly compare the service upper-bound times, $\mathcal{T}_{\mathcal{UB}}$, calculated in Eq. 23, 24 and 28, with the remaining battery-life of action agent M at time t, by updating $\mathcal{T}_{\mathcal{UB}}$ as $\mathcal{T}_{\mathcal{UB}} \leftarrow$ minimum $\left(\mathcal{T}_{\mathcal{UB}}, T_t^{B_M}\right)$. The output of this update will be used in Eq. 21 to propagate the uncertainty of the target locations at most as long as it will take the action agent to provide service or its battery-life allows it to operate.

VI. Empirical Evaluation

In this section, we empirically evaluate both our high-level decision making and low-level coordinated control modules in an aerial wildfire fighting case-study, in which teams of heterogeneous, autonomous UAVs are tasked to work together to fight a propagating wildfire efficiently, with limited resources, such as the number of available UAVs and battery-lives. Moreover, we perform several experiments to evaluate the scalability and computational efficiency of our framework as well as its compatibility to different exploration strategies.

In this particular simulation, we generated ten different terrains of 100×100 with 30 initial firespots of different intensities. The firespots move and propagate according to the FARSITE model introduced in Section II-A. The parameters of FARSITE model R_t (spread rate), U_t (wind velocity) and θ_t (wind azimuth) are initialized as normally distributed random variables with $\langle \mu_R = 6, \sigma_R = 3 \rangle$, $\langle \mu_U = 5, \sigma_U = 2 \rangle$ and $\langle \mu_\theta = \pi/4, \sigma_\theta = 1 \rangle$, respectively.

Each UAV, d, is assigned a linear velocity range of $[v_{min}^d, v_{max}^d]$ and a maximum angular velocity, ω_{max}^d. Each class of UAVs (e.g., fixed-wing or quadcopter) has its own special characteristic, for instance for fixed-wing UAVs $v_{min} \neq 0$ while for quadcopters, $v_{min} = 0$ and $\omega_{max} = \infty$. All mobile robots in this environment are modeled as Dubins vehicles. In our experiments, we only assign quadcopters to perception agents and fixed-wing aircraft to action agents.

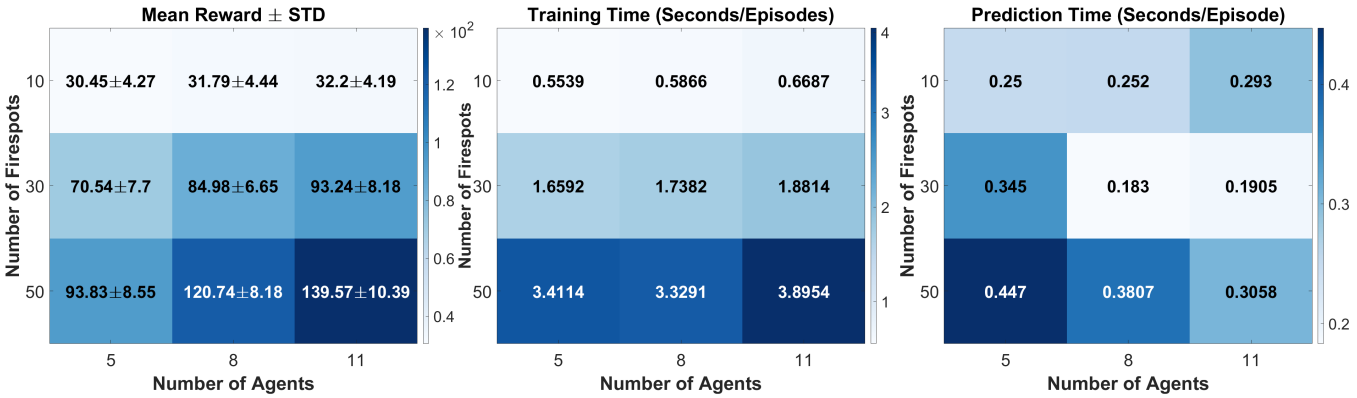


Figure 9: This figure depicts our scalability and computation time results. The leftmost figure shows the mean reward accumulated in nine different scenarios specified by the number of agents in the composite team and the number of firespots in the environment. Test repeated for 10 times and standard deviations are presented (± STD). The middle and rightmost charts depict the training and prediction times for the same nine scenarios, respectively. The times are shown as seconds per episodes.

Each perception UAV is controlled by a shared NN trained via our MA-SARTSA learning algorithm (Section IV-E). The policy network architecture consists of two fully connected layers (hidden units are 128 and 32, respectively) with ReLU activations. Fire intensity increases if a firespot is left unexploited. The NN represents the Q-function, $Q_\theta : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$, inputting observation and action and outputting the corresponding Q-value. For discrete action spaces, we obtain the policy via $\pi = \arg\max_{a \in \mathcal{A}} Q_\theta(o, a)$. If a firespot is exploited (i.e., a perception agent travels to it, extracts its state information, generates a guaranteed trajectory for action agent, and an action agent executes that trajectory), fire intensity drops as a result of the action agent's firefighting service. Rewards, which are equal to the initial fire intensity, will only be given when the fire is extinguished. The learning rate, α, temporal discount factor, γ, and training epoch number were chosen empirically to be 0.001, 0.999 and 6000, respectively. Fig. 8 (right) demonstrates the RL agent's cumulative reward throughout training and indicates success.

A. Benchmarks

We compared our approach with two categories of heuristic benchmarks: (1) ϵ-greedy and (2) myopic, as described below:

- **ϵ-greedy:** The perception agents on the composite team exploit a firespot with probability ϵ and choose a known firespot according to uniform distribution. With probability $1-\epsilon$, agents choose to explore the terrain to discover new firespots. As an extreme case, when $\epsilon = 1$, agents will only explore when there is no known fire to exploit.
- **Myopic:** The perception agents on the composite team always exploit a fire immediately after discovering it until the fire is extinguished by the action agent. When the current fire is extinguished, the agent will choose to explore. Myopic agents do not cooperate as each agent only exploits a fire that the agent has found.

B. Evaluation Results

To evaluate the performance against benchmarks, we considered three metrics: (1) the discounted cumulative reward (our learning algorithm's objective), (2) the time required for agents to find all the firespots, and (3) the time required for the team to extinguish all firespots (i.e., the perception-action cooperation objective). The results of the evaluation over 100 trials of simulation are presented in Fig. 8. For all three evaluation metrics, our approach outperforms the baselines.

We performed statistical tests to show our approach's superior performance on all three metrics. We tested for normality and homoscedasticity and did not reject the null hypothesis for rewards and time to find all fires using Shapiro-Wilk ($p > .2$) and Levene's Test ($p > .2$), but we reject normal hypothesis for time to extinguish all fires. One-way ANOVA showed a significant difference between four groups on discounted cumulative rewards earned and time to find all fires ($p < .001$). Tukey's posthoc tests showed a significant difference between our approach and all benchmarks. Kruskal-Wallis test showed a significant difference between the four groups on time to extinguish all fires ($p < .001$), and posthoc tests showed also showed significant improvements.

Scalability and Sensitivity Analysis – In our scalability experiments, we tested the performance of our MA-SARTSA algorithm over various environmental and algorithmic parameters. We tested the scalability of our algorithm on a set of different number of agents in the composite team (5, 8 and 11 agents) and number of propagating firespots in the terrain (10, 30 and 50 firespots). We computed the accumulated mean reward (± Standard Deviation (STD)) as well as the training and prediction times across nine different environment setups. The result are represented in Fig. 9. As shown, the training time increases as the size of fire and number of agents increase, while the prediction time decreases when the number of agents increases. Moreover, the accumulated average reward also increases significantly when the number of agents and firespots increase. Accordingly, our framework scales to the number of firespots and expectedly can achieve higher performance when the number of agents increases. We also tested the sensitivity of our framework to different values of agents' FOV size as a key parameter, which has a direct correlation with the $\Delta-$disk proximity in our CE-TSP approach. Again, we evaluated the performance by computing the mean rewards (±STD) and computation times as described above, on our original environment setup described in Section VI. As represented in

Table II: Results of our Sensitivity experiment for different values of agents' FOV sizes. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$.

FOV	Reward ($\mu\pm$std)	Training Time	Prediction Time
1×1	69.97±6.12	1.7816	0.187
3×3	70.38±5.37	1.7962	0.36
5×5	70.06±7.07	1.8300	0.326

Table III: Results of our performance sensitivity evaluation under horizontal sweeping, diagonal sweeping and spiral exploration paths. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$.

Method	Reward ($\mu\pm$std)	Training Time	Prediction Time
Horizontal	69.97±6.12	1.7816	0.187
Diagonal	61.46±7.34	1.7882	0.537
Spiral	68.21±7.95	1.9432	0.445

Table II, by increasing the size of agents' FOVs from a 1×1 grid (e.g., only an agent's location) to a 5×5 grid (e.g., two-hop neighborhoods around agents), our framework is capable of achieving similar results and thus, shows low sensitivity to values of FOV and $\Delta-$disk.

We also evaluated our framework under different exploration approaches: (1) horizontal sweeping, (2) diagonal sweeping and (3) spiral paths. The goal was to investigate the sensitivity of the framework to the exploration paths taken by agents. As in our environment setup from Section VI (e.g., five agents and 30 initial firespots), we computed the total average rewards, as well as the training and prediction times for comparison between exploration methods, as shown in Table III. Our framework achieves similar performances in terms of average total reward and computation times under different exploration methods, demonstrating low sensitivity to the exploration approach.

VII. Demonstration: Multi-Robot Testbed

We evaluated our coordination module on physical robots to directly include robot's motion dynamics and to test the calculated timings. The coordinated control and planning module was implemented and tested in the Robotarium multi-robot platform [30] on two simple (dynamic targets with linear motion models) and complex (propagating wildfire with numerous dynamic firespots) scenarios. Each case is elaborated below.

Demonstration Scenario (A) - Dynamic Targets with Linear Motion Models:

For this case, six dynamic targets (one target is intentionally left stationary, $\dot{q} = 0$) are randomly initialized in a terrain. Targets are assumed to have a linear motion model (moving on a straight line) with constant velocities, and their locations are initially unknown to four robots. The composite robot team includes two perception and two action robots that are required to stop the targets from moving. Accomplishing this task can be achieved by moving an action robot to the location of the respective target; however, since action agents are unable to see targets, a collaboration between perception and action robots is required. We apply our coordination framework to enable this collaboration. Excerpts of this experiment are depicted and described in Fig. 10. As a result, our composite robot team successfully discovered (by perception robots) and stopped (by action robots) all six initially unknown targets collectively, without losing the track of any targets. The video recording of this experiment can be found in the first part of the provided supplementary video as well as at `https://youtu.be/qcomKuD-Hhw`.

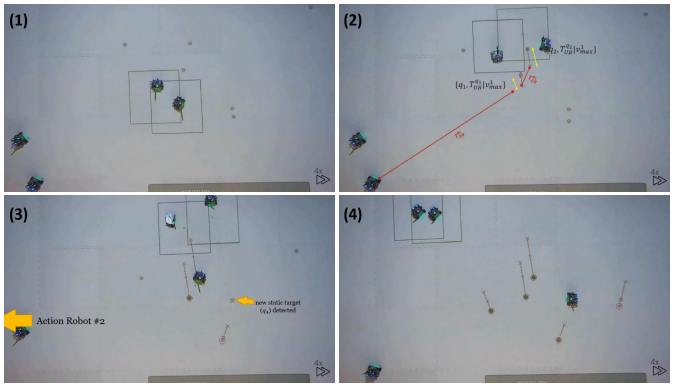


Figure 10: (1) Two perception robots start exploring the environment for targets. (2) Perception robots evaluate the TEB bound for an action robot. (3) The action robot starts the task the first time TEB is violated; perception robots continue to explore. (4) Final standing of the robot team after servicing all six initially unknown targets.

Demonstration Scenario (B) - Propagating Wildfire with Numerous Dynamic Firespots:

In this case, we evaluated our perception-action robot team coordination module in a more complex aerial wildfire-fighting environment. Four randomly initialized distinct fire areas are considered to be explored by two perception robots. FARSITE wildfire propagation model has been leveraged, and the model parameters are initialized as detailed in Section VI. We note that in our experiments we use omnidirectional robots; as such, we do not need to leverage our scanning framework (Section V-B) and perception agents pass propagated centroid nodes of discovered groups of firespots to three action robots. The video recording of this experiment can be found in the second part of the provided supplementary video as well as at `https://youtu.be/qcomKuD-Hhw`. For comparison, an embedded simulated video is added on the top-left of the screen to demonstrate the wildfire propagation simulation with similar parameters but without the composite team attempting to fight the wildfire. As a result, our composite robot team successfully performed the complex task of autonomous wildfire fighting by effectively coordinating for putting out all firespots.

VIII. Discussion

Our empirical evaluations demonstrate the feasibility of our framework and its superior performance in all three evaluation metrics stated in Section VI-B, as compared to the employed benchmarks. For the epsilon-greedy agent, we tested various epsilons, $\epsilon \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ where $\epsilon = 0.8$ obtained the best result among all the ϵ-greedy agents; yet, our MA-SARTSA agent outperformed the 0.8-greedy agent, as well as the greedy agent (i.e., 1-greedy). We can see from the middle bar-plot in Fig. 8 that the 0.8-greedy agent spends slightly less time to find all of the fire spots as compared to

the Myopic agent, while on the other hand, the Myopic agent spends slightly less time to extinguish all of the fires than the 0.8-greedy agent. Accordingly, none of the Myopic or ϵ-greedy policies can be concluded to be an empirically better policy than the other. The MA-SARTSA learning algorithm successfully helps the policy to maximize the objective in Eq. 4 as is shown in the learning curve in Fig. 8 (right). The figure shows that within 2000 episodes, the algorithm has already been able to find a much better policy than its initial policy (random), illustrating the empirical success of MA-SARTSA learning and sample efficiency.

Nevertheless, the significant improvement of our MA-SARTSA framework over both the Myopic and the ϵ-greedy agents shows that the perception agents must make decisions based on not only the state information acquired from the environment, but also according to their collaborator action-agents' characteristics and motion dynamics. Myopic and ϵ-greedy agents do not model the time it takes for action agents to finish their tasks. Our MA-SARTSA, however, incorporates action agents' upper-bound service times (e.g., the time it takes for action agents to finish their tasks), calculated in the lower-level coordinated control module. Moreover, in our coordinated control and planning framework, we explicitly reason about the number of targets assigned to an action agent based on the respective action agent's dynamics and motion restrictions, such as maximum velocity. The effective performance of our perception-action collaboration and the described framework is empirically validated in our demonstrations and physical-robot experiments (presented in Section VII).

Note: We assume that agents do not fail (e.g., UAV failure) and that the underlying motion model of the targets is known. In order to account for model deviations, we leverage an EKF-based system to estimate the parameters of this model, resulting in a measurement uncertainty based analytical tracking error bound (TEB), introduced in Eq. 21. Accordingly, three parameters, the process and observation noise covariances, Q_t and Γ_t, as well as the $\mathcal{E}$ in TEB, can be used to tune the reliability of the utilized motion model. If the assumed target motion model is accurate but the UAV sensors are noisy, Q_t and Γ_t can be initialized with small and large values to put more weight on the model and less on the UAV observation model, respectively (and vice versa). Moreover, the $\mathcal{E}$ in TEB can be tuned to more conservative values (e.g., larger values) for when the underlying process and observation models are less accurate, and vice versa.

In our low-level coordinated control framework we leveraged the CE-TSP for perception agents to account for their FOV; however, we note that it is an application-specific consideration to decide whether the Action agents can actually actuate at a certain distance with respect to targets. While in some applications the exact location of a moving target may be required, in many other cases such as our wildfire fighting case-study, oil-spill control in oceans or search-and-rescue, there exist numerous static or dynamic targets where action agents can actuate (e.g., dump water extinguisher on fire) *close-enough* to target locations. In such cases, as discussed in Section V-A1, our EKF-based system provides the MMSE estimate of the posterior which is assumed to be acceptable. Moreover, as noted above, the hyper-parameter $\mathcal{E}$ in Eq. 21 can also be tuned based on the background application to improve the tracking accuracy to a satisfying threshold. Nevertheless, in our framework, the accuracy of the action agents moving to the exact target locations remains dependent on the model and measurement accuracy as well as well-tuning Eq. 21.

IX. Conclusion

In this paper, we have introduced a novel hierarchical approach to jointly tackle the high-level decision making and low-level collaborative control problems for heterogeneous teams of autonomous robots consisting of perception agents and action agents. In our centralized high-level decision-making module, we propose MA-SARTSA-based learning under our MA-POSMDP model to enable perception agents to explore an unknown environment (i.e., discover dynamic targets) and exploit known targets by extracting their state information. Extracted state information is required for action agents (which are unable to sense targets) for manipulation.

We have also introduced a measurement-uncertainty based tracking error and derived a set of analytical upper-bound service times to ensure a probabilistically guaranteed service for action agents in various scenarios. These upper-bound service times are leveraged into our high-level decision-making module for task timings in the introduced MA-POSMDP model. Additionally, we introduced a coordinated routing problem with an attribute-based robot-interaction scheme through which the perception-action agents cooperation is individualized to account for robots heterogeneity and improve the composite team's resiliency and performance.

Finally, we motivated the application of perception-action composite robot teams for autonomous aerial wildfire fighting and created a simulated environment in which our quantitative evaluations validate the performance and efficiency of our frameworks by reducing the overall operation times.

Appendix

A. Proof of Theorem 2 (Leveraging Wildfire Propagation and UAV Observation Models)

We first derive the process and observation Jacobian matrices in EKF's state transition and observation equations and their derivatives with respect to state variables. Considering Eq. 10-13 and Fig. 4 as the UAV perception model, the state transition Jacobian matrix F_t can be represented as in Eq. 30, where $t' = t-1$ and superscript (3) represent the number of column and row repetitions for all $[p_t^x, p_t^y, p_t^z]$.

$$\left.\frac{\partial \mathcal{M}_t}{\partial \mathcal{S}_i}\right|_{\hat{\mathcal{S}}_{t'}} = \begin{array}{c} \\ q_t^x \\ q_t^y \\ p_t^{(3)} \\ R_t \\ U_t \\ \theta_t \end{array} \begin{array}{c} \begin{array}{cccccc} q_{t'}^x & q_{t'}^y & p_{t'}^{(3)} & R_{t'} & U_{t'} & \theta_{t'} \end{array} \\ \begin{pmatrix} 1 & 0 & 0^{(3)} & \frac{\partial q_t^x}{\partial R_{t'}} & \frac{\partial q_t^x}{\partial U_{t'}} & \frac{\partial q_t^x}{\partial \theta_{t'}} \\ 0 & 1 & 0^{(3)} & \frac{\partial q_t^y}{\partial R_{t'}} & \frac{\partial q_t^y}{\partial U_{t'}} & \frac{\partial q_t^y}{\partial \theta_{t'}} \\ 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} \\ 0 & 0 & 0^{(3)} & 1 & 0 & 0 \\ 0 & 0 & 0^{(3)} & 0 & 1 & 0 \\ 0 & 0 & 0^{(3)} & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{30}$$

Note that despite the subscript t parameters R_t, U_t, and θ_t are not necessarily dynamic with time, and it is fairly reasonable to consider these physical parameters as constants for short periods of time, since we assume locality in time and space according to FARSITE [31]. To calculate the partial derivatives in Eq. 30, we need the fire dynamics presented by FARSITE wildfire mathematical model in Section II-A. Now, the derivatives of q_t^x and q_t^y with respect to parameters R_{t-1}, U_{t-1}, and θ_{t-1} are computed by applying the chain-rule and using Eq. 1-2 as follows in Eq. 31-33, where $\mathcal{D}(\theta)$ equals $\sin\theta$ and $\cos\theta$ for X and Y axis, respectively.

$$\frac{\partial q_t}{\partial \theta_{t-1}} = C(R_t, U_t)\frac{\partial \mathcal{D}(\theta)}{\partial \theta}\delta t \tag{31}$$

$$\frac{\partial q_t}{\partial R_{t-1}} = \left(1 - \frac{LB(U_t)}{LB(U_t) + \sqrt{GB(U_t)}}\right)\mathcal{D}(\theta)\delta t \tag{32}$$

$$\frac{\partial q_t}{\partial U_{t-1}} = \frac{R_{t'}\left(LB(U_{t'})\frac{\partial GB(U_{t'})}{\partial U_{t'}} - GB(U_{t'})\frac{\partial LB(U_{t'})}{\partial U_{t'}}\right)}{\left(LB(U_{t'}) + \sqrt{GB(U_{t'})}\right)^2}\mathcal{D}(\theta)\delta t \tag{33}$$

Accordingly, the observation Jacobian matrix H_t can be represented as follows in Eq. 34 where $t' = t - 1$.

$$\left.\frac{\partial \mathcal{O}_t}{\partial \Phi_i}\right|_{\hat{\Phi}_{t'}} = \begin{array}{c} \\ \varphi_t^x \\ \varphi_t^x \\ \hat{R}_t \\ \hat{U}_t \\ \hat{\theta}_t \end{array} \begin{array}{c} \begin{array}{ccccccccc} q_t^x & q_t^y & p_t^x & p_t^y & p_t^z & R_t & U_t & \theta_t \end{array} \\ \begin{pmatrix} \frac{\partial \varphi_t^x}{\partial q_t^x} & 0 & \frac{\partial \varphi_t^x}{\partial p_t^x} & 0 & \frac{\partial \varphi_t^x}{\partial p_t^z} & 0 & 0 & 0 \\ 0 & \frac{\partial \varphi_t^y}{\partial q_t^y} & 0 & \frac{\partial \varphi_t^y}{\partial p_t^y} & \frac{\partial \varphi_t^y}{\partial p_t^y} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{34}$$

The angle parameters (i.e., φ_t^x and φ_t^y) contain information regarding both firefront location $[q_t^x, q_t^y]$ and UAV coordinates $[p_t^x, p_t^y, p_t^z]$. According to Fig. 4, by projecting the looking vector of UAV to planar coordinates, the angle parameters are calculated as $\varphi_t^x = \tan^{-1}\left(\frac{p_t^z}{\|q_t - p_t\|}\right)$ and $\varphi_t^y = \tan^{-1}\left(\frac{\|q_t - p_t\|}{p_t^z}\right)$ for X and Y axes respectively, where $q_t = [q_t^x, q_t^y]$ and $p_t = [p_t^x, p_t^y]$. Now, the partial derivatives in the observation Jacobian matrix H_t for *X*-axis, presented in Eq. 34, are derived as in Eq. 35-37 and for *Y*-axis derivatives, we can derive as in Eq. 38-40.

$$\nabla_{q_t}\varphi_t^x = \frac{1}{1 + \left(\frac{p_t^z}{\|q_t - p_t\|}\right)^2}\left(\frac{-p_t^z\,(q_t - p_t)}{\|q_t - p_t\|^3}\right) = \left[\frac{\partial \varphi_t^x}{\partial q_t^x}, \frac{\partial \varphi_t^x}{\partial q_t^y}\right] \tag{35}$$

$$\nabla_{p_t}\varphi_t^x = \frac{1}{1 + \left(\frac{p_t^z}{\|q_t - p_t\|}\right)^2}\left(\frac{p_t^z\,(q_t - p_t)}{\|q_t - p_t\|^3}\right) = \left[\frac{\partial \varphi_t^x}{\partial p_t^x}, \frac{\partial \varphi_t^x}{\partial p_t^y}\right] \tag{36}$$

$$\frac{\partial \varphi_t^x}{\partial p_t^z} = \frac{1}{1 + \left(\frac{p_t^z}{\|q_t - p_t\|}\right)^2}\left(\frac{1}{\|q_t - p_t\|}\right) \tag{37}$$

$$\nabla_{q_t}\varphi_t^y = \frac{1}{1 + \left(\frac{\|q_t - p_t\|}{p_t^z}\right)^2}\left(\frac{(q_t - p_t)}{p_t^z\,\|q_t - p_t\|}\right) = \left[\frac{\partial \varphi_t^y}{\partial q_t^x}, \frac{\partial \varphi_t^y}{\partial q_t^y}\right] \tag{38}$$

$$\nabla_{p_t}\varphi_t^y = \frac{1}{1 + \left(\frac{\|q_t - p_t\|}{p_t^z}\right)^2}\left(\frac{-(q_t - p_t)}{p_t^z\,\|q_t - p_t\|}\right) = \left[\frac{\partial \varphi_t^y}{\partial p_t^x}, \frac{\partial \varphi_t^y}{\partial p_t^y}\right] \tag{39}$$

$$\frac{\partial \varphi_t^y}{\partial p_t^z} = \frac{1}{1 + \left(\frac{\|q_t - p_t\|}{p_t^z}\right)^2}\left(\frac{-\|q_t - p_t\|}{(p_t^z)^2}\right) \tag{40}$$

Moreover, the model and observation measurement uncertainties associated with EKF estimation follow the general nonlinear uncertainty propagation law, shown in Eq. 19-20. Accordingly, when stationary, the uncertainty can be propagated and updated as in Eq. 41, where $K_t = \Sigma_{t|t-1}H_t^T\Lambda_t^{-1}$ is the near-optimal Kalman gain and $\tilde{y}_t = z_t - h(\hat{\Lambda}_{t|t-1})$ is the measurement residual in which $h(.)$ is the observation model.

$$\hat{\Lambda}_{t|t} = \hat{\Lambda}_{t|t-1} + K_t\tilde{y}_t \tag{41}$$

Now, considering EKF's covariance propagation equations in Eq. 19-20 and 41 as well as the gradients in process Jacobian matrix F_t as calculated in Eq. 31-33 and the gradients in observation Jacobian matrix H_t as calculated in Eq. 35-37, we can readily see that the gradients in process Jacobian matrix (Eq. 30) are only functions of fire propagation model parameters such as fuel coefficient R_t, mid-flame wind velocity U_t, wind azimuth and θ_t. Consequently, while these parameters do not vary significantly with time, the uncertainty drop due to process model is time-invariant. We note that FARSITE [31] assumes locality in time (i.e., within seconds or few minutes), making the assumption of time-invariant fire parameters fairly acceptable. This is also acceptable physically to assume that wind velocity or vegetation coefficient stays constant within time-intervals of seconds or a few minutes [56]. Additionally, the gradients in the observation Jacobian matrix (i.e., Eq. 35-37 and Eq. 34) are only functions of the Euclidean distance between the UAV position and fire-spot location. At the time of visiting a fire spot the planar displacement between UAV and fire locations are zero and the only distance between the two is the UAV altitude. Accordingly, both F_t and H_t are time-invariant and with constant process and observation noise covariances (Q_t and Γ_t), the total uncertainty drop as propagated by EKF is not a function of time nor planar displacement, and thus is time-invariant.

B. Derivation of the Escape Upper-bound Time $\mathcal{T}_{max}^l$ in Eq. 29 (Case 3)

For this case we follow the derivations in our previous work [24]. Considering the case detailed in Section V-D3, for a specific fire, q_t, the time-varying width, $w(t)$, and height (i.e., planar length), $h(t)$, of the enlarging fire area can be estimated as $w(t) \leq 2\,\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}$ and $h(t) \leq 2\,\widehat{\dot{q}_t^y}\Big|_\alpha T_{UB}^{(C3)}$ at the α confidence level, as fires are now allowed to spread for $T_{UB}^{(C3)}$ units of time. Assuming a vertical scanning pattern, we round up the maximum possible $w(t)$ and thus, the number of passes the UAV with the FOV of width $\mathcal{W}_t$ would take from left to right is given by $n(t) \leq \left\lceil \frac{2\,\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil$. The total distance

traversed for each pass is $d(t) = 2\,\widehat{\dot{q}_t^{y}}\Big|_{\alpha} T_{UB}^{(C3)}$ and finally the total pass traversed can be calculated as $d^{tot}(t) = n(t)h(t)$. Now, the time required for one firefront point to escape the FOV of a specific UAV d with velocity v can be calculated as

$$\tau_q(t) = \frac{d^{tot}(t)}{v} = \frac{n(t)h(t)}{v} = \frac{\left\lceil \frac{2\,\widehat{\dot{q}_t^{x}}\Big|_{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil 2\,\widehat{\dot{q}_t^{y}}\Big|_{\alpha} T_{UB}^{(C3)}}{v} \tag{42}$$

Now, the time required for all of the fire points will be the summation over all τ_q in Eq. 42. However, we must note that the center of each spreading fire point also moves, and the upper bound time derived for case 2 (i.e., $T_{UB}^{(C2)}$) must be added to this summation. Therefore, the upper bound time for case 3 can be estimated as in Eq. 43.

$$T_{UB}^{(C3)} = T_{UB}^{(C2)} + \sum_{q} \frac{2}{v}\,\widehat{\dot{q}_t^{y}}\Big|_{\alpha} T_{UB} \left\lceil \frac{2\,\widehat{\dot{q}_t^{x}}\Big|_{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil \tag{43}$$

We need to solve Eq. 43 for $T_{UB}^{(C3)}$ to find the final equation to obtain the upper bound. For this purpose, we make the following two simplifying assumptions. First, we remove the ceiling operator and add one to the term inside the operator to achieve continuity, as shown in Eq. 44.

$$\left\lceil \frac{2\,\widehat{\dot{q}_t^{x}}\Big|_{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil \leq \frac{2\,\widehat{\dot{q}_t^{x}}\Big|_{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} + 1 \tag{44}$$

Second, we assume the area required to cover to account for the growth of each fire location is upper-bounded by the size of the generated waypoint list $|\hat{s}_t^l|$ times the area of growth for a hypothetical fire growing quickest (Eq. 45).

$$\sum_{q} \widehat{\dot{q}_t^{y}}\Big|_{\alpha} T_{UB} \left\lceil \frac{2\,\widehat{\dot{q}_t^{x}}\Big|_{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil \leq |\hat{s}_t^l|\zeta^{\alpha} \left\lceil \frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} \right\rceil \tag{45}$$

With these two conservative simplifying assumptions, the bound in Eq. 43 can be revised as below.

$$T_{UB}^{(C3)} = T_{UB}^{(C2)} + \frac{2|\hat{s}_t^l|\zeta^{\alpha} T_{UB}^{(C3)}}{v} \left[\frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} + 1 \right] \tag{46}$$

$$T_{UB}^{(C3)} - \frac{2|\hat{s}_t^l|\zeta^{\alpha} T_{UB}^{(C3)}}{v} \left[\frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} + 1 \right] = T_{UB}^{(C2)} \tag{47}$$

$$T_{UB}^{(C3)} - \frac{2|\hat{s}_t^l|\zeta^{\alpha} T_{UB}^{(C3)}}{v} \left[\frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{\mathcal{W}_t} + 1 \right] = \frac{\Delta L_t}{\frac{v}{2} - 2\zeta^{\alpha}(|\hat{s}_t^l| - 1)}$$

in which $\Delta L_t = \sum_{n=0}^{l-1} \left\| q_t^{n+1} - q_t^{n} \right\|$ is the total travel distance for action agent between the last node added to the waypoint set and the first node. Note that the Eq. for $T_{UB}^{(C3)}$ is in the form of $z - az\,(bz + 1) = \delta$ general quadratic equations with $z = T_{UB}^{(C3)}$, which can be simplified to $\gamma z^2 - \beta z + \delta = 0$, where $\gamma = ab$ and $\beta = 1 + a$, and they can be calculated as $\gamma = \frac{4|\hat{s}_t^l|(\zeta^{\alpha})^2}{\mathcal{W}_t v}$, $\beta = 1 + \frac{2|\hat{s}_t^l|\zeta^{\alpha}}{v}$ and $\delta = \frac{2\Delta L_t}{v\left(1 - 2\zeta^{\alpha}(|\hat{s}_t^l| - 1)\right)}$. Eventually, from the solution to general quadratic equations, we know that the upper-bound service time $T_{UB}^{(C3)}$ for action agents in Case 3 can be obtained as in Eq. 48.

$$T_{UB}^{(C3)} = \frac{-\beta + \sqrt{\beta^2 - 4\gamma\delta}}{2\gamma} \tag{48}$$

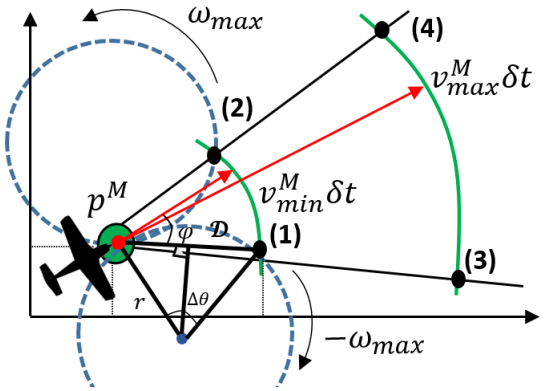


Figure 11: Vertex coordinates of the reachable polygon, introduced in Fig. 6, are calculated to generate fixed-wing-friendly paths.

C. Calculating Coordinates of Reachable Polygon's Vertices in Scanning Framework

Fig. 11 is presented to elaborate on the calculation of coordinates enclosing the reachable areas by the manipulator agent. To calculate the X-Y coordinates of the four vertices, shown in Fig. 11, we first need to form the reachable polygon. As such, we draw two circles centered at the agent's current position (green arches in Fig. 11) with radius $\mathcal{D}_{min} = v_{min}^{M}\delta$ and $\mathcal{D}_{max} = v_{max}^{M}\delta t$. $\mathcal{D}_{min}$ and $\mathcal{D}_{max}$ are the agent's lower and upper-bound planar displacement for one unit of time, δt, if the agent is moving with its minimum or maximum velocity. We call these two circles, the *planar displacement circles*. Next, we need to account for the action agent's angular velocity, ω_{max} (e.g., turning bank). Accordingly, we draw two more circles (blue dashed-circles in Fig. 11) from the agent's current location in clockwise and counter-clockwise directions. We refer to these two circles as *angular motion circles*. The area enclosed by the intersections of these four circles is the *reachable* area by the action agent, M, with velocity range of $\left[v_{min}^{M}, v_{max}^{M}\right]$ and maximum turning bank of ω_{max}.

Accordingly, the desired coordinates of the four vertices of the reachable polygon can be obtained by projecting the agent's current XY coordinates according to the joint angular-planar rotation mappings, as described in Section V-B. In Eq. 18, $(-1)^i$ is to account for the clockwise and counter-clockwise directions of the angular velocity, ω_{max}, and $R_z^{\varphi}(\varphi)$ is the rotation matrix around the z-axis as in Eq. 49.

$$R_z^{\varphi}(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \tag{49}$$

To calculate the angle, φ, according to Fig. 11, we consider the velocity-vector tangent to the angular motion circle. The planar position rotation angle $\Delta\theta$ can be calculated as $\Delta\theta = \omega_{max}\delta t$, and therefore, the angle between the tangent velocity vector and planar displacement line, φ, can be calculated as $\varphi = \frac{\Delta\theta}{2} = \frac{\omega_{max}\delta t}{2}$. The proposed scanning framework is of particular interest for action robots that have restricted motions (i.e., fixed-wing UAVs with non-zero minimum velocity, $v_{min} \neq 0$, and $\omega \in [-\omega_{max}, \omega_{max}]$). In such cases, our coordinated scanning framework leads to action agents being able to directly traverse between any two points inside the reachable polygons with near-linear approximation. In the

case of omni-directional robots (e.g., multi-rotor UAVs), the proposed scanning approach can be ignored.

REFERENCES

[1] H. Ravichandar, K. Shaw, and S. Chernova, "Strata: unified framework for task assignments in large teams of heterogeneous agents." *(JAAMAS)*, vol. 34, no. 2, p. 38, 2020.

[2] E. Seraj and M. Gombolay, "Coordinated control of uavs for human-centered active sensing of wildfires," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1845–1852.

[3] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *(CSUR)*, vol. 52, no. 2, p. 29, 2019.

[4] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *(IJRR)*, vol. 32, no. 12, pp. 1495–1512, 2013.

[5] E. Seraj, X. Wu, and M. Gombolay, "Firecommander: An interactive, probabilistic multi-agent environment for joint perception-action tasks," *arXiv preprint arXiv:2011.00165*, 2020.

[6] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad hoc networks*, vol. 2, no. 4, pp. 351–367, 2004.

[7] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of Applied Probability*, vol. 25, no. A, pp. 287–298, 1988.

[8] H. Yuan, H. Ma, and H. Liao, "Coordination mechanism in wireless sensor and actor networks," in *(IMSCCS'06)*, vol. 2. IEEE, 2006, pp. 627–634.

[9] N. Sabri, S. Aljunid, R. Ahmad, M. Malik, A. Yahya, R. Kamaruddin, and M. Salim, "Wireless sensor actor networks," in *(ISWTA)*. IEEE, 2011, pp. 90–95.

[10] K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, "Distributed recovery of actor failures in wireless sensor and actor networks," in *(WCNC)*. IEEE, 2008, pp. 2480–2485.

[11] A. Alfadhly, U. Baroudi, and M. Younis, "Least distance movement recovery approach for large scale wireless sensor and actor networks," in *(IWCMC)*. IEEE, 2011, pp. 2058–2063.

[12] M. J. Bays and T. A. Wettergren, "A solution to the service agent transport problem," in *(IROS)*. IEEE, 2015, pp. 6443–6450.

[13] F. Xia, Y.-C. Tian, Y. Li, and Y. Sung, "Wireless sensor/actuator network design for mobile control applications," *Sensors*, vol. 7, no. 10, pp. 2157–2173, 2007.

[14] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 964–973.

[15] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "A distributed coordination framework for wireless sensor and actor networks," in *(ACM Mobihoc)*. ACM, 2005, pp. 99–110.

[16] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," *(JFR)*, vol. 23, no. 3-4, pp. 165–184, 2006.

[17] E. Seraj, v. Azimi, C. Abdallah, S. Hutchinson, and M. Gombolay, "Adaptive leader-follower control for multi-robot teams with uncertain network structure," in *2021 American Control Conference (ACC)*. IEEE, 2021.

[18] Z. Wang and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints." RSS, 2020.

[19] H. Momeni, M. Sharifi, and S. Sedighian, "A new approach to task allocation in wireless sensor actor networks," in *(CICSN)*. IEEE, 2009, pp. 73–78.

[20] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[21] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative control: models, applications and algorithms*. Springer, 2003, pp. 23–41.

[22] P. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple UAVs," in *(CDC)*. IEEE, 2007, pp. 4875–4880.

[23] K. Sundar, S. Venkatachalam, and S. G. Manyam, "Path planning for multiple heterogeneous unmanned vehicles with uncertain service times," in *(ICUAS)*. IEEE, 2017, pp. 480–487.

[24] E. Seraj, A. Silva, and M. Gombolay, "Safe coordination of human-robot firefighting teams," *arXiv preprint arXiv:1903.06847*, 2019.

[25] M. Y. Sir, I. F. Senturk, E. Sisikoglu, and K. Akkaya, "An optimization-based approach for connecting partitioned mobile sensor/actuator networks," in *(INFOCOM WKSHPS)*. IEEE, 2011, pp. 525–530.

[26] Y.-L. Lai and J.-R. Jiang, "Optimal path planning for fault-tolerant and energy-efficient target surveillance in wireless sensor and actor networks," in *(MDM: Systems, Services and Middleware)*. IEEE, 2009, pp. 531–535.

[27] J. Le Ny, M. Dahleh, and E. Feron, "Multi-UAV dynamic routing with partial observations using restless bandit allocation indices," in *(ACC)*. IEEE, 2008, pp. 4220–4225.

[28] S. Guha, K. Munagala, and P. Shi, "Approximation algorithms for restless bandit problems," *(JACM)*, vol. 58, no. 1, p. 3, 2010.

[29] M. Baykal-Gürsoy, "Semi-markov decision processes," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[30] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *(ICRA)*. IEEE, 2017, pp. 1699–1706.

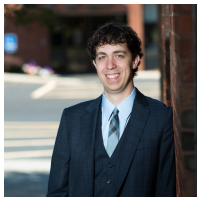
[31] M. A. Finney, "Farsite: Fire area simulator-model development and evaluation," *Res. Pap. RMRS-RP-4. Ogden, UT: US Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p.*, vol. 4, 1998.

[32] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A

distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," in *(IROS)*. IEEE, 2017, pp. 6648–6653.
[33] M. E. Alexander, "Calculating and interpreting forest fire intensities," *Canadian Journal of Botany*, vol. 60, no. 4, pp. 349–357, 1982.
[34] D. Bertsimas and J. Niño-Mora, "Restless bandits, linear programming relaxations, and a primal-dual index heuristic," *Operations Research*, vol. 48, no. 1, pp. 80–90, 2000.
[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *(ICML)*, 2016, pp. 1928–1937.
[36] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[37] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *(IROS)*. IEEE, 2018, pp. 1067–1074.
[38] Y. Hsu, H. Wu, K. You, and S. Song, "A selected review on reinforcement learning based control for autonomous underwater vehicles," *arXiv preprint arXiv:1911.11991*, 2019.
[39] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen, "Reinforcement learning for autonomous UAV navigation using function approximation," in *(SSRR)*. IEEE, 2018, pp. 1–6.
[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[41] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the kalman filter," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 7, pp. 37–38, 2004.
[42] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive Kalman filter," *(JVCIR)*, vol. 17, no. 6, pp. 1190–1208, 2006.
[43] H. A. Patel and D. G. Thakore, "Moving object tracking using kalman filter," *(IJCSMC)*, vol. 2, no. 4, pp. 326–332, 2013.
[44] R. K. Ramachandran, N. Fronda, and G. S. Sukhatme, "Resilience in multi-robot multi-target tracking with unknown number of targets through reconfiguration," *arXiv preprint arXiv:2004.07197*, 2020.
[45] G. Welch, G. Bishop *et al.*, "An introduction to the Kalman filter," 1995.
[46] P. Newman, "Ekf based navigation and slam, slam summer school 2006," 2006.
[47] M. Reinštein, "From bayes to extended kalman filter."
[48] D. Simon, *Optimal state estimation: kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
[49] M. Di, E. M. Joo, and L. H. Beng, "A comprehensive study of kalman filter and extended kalman filter for target tracking in wireless sensor networks," in *(IEEE SMC)*. IEEE, 2008, pp. 2792–2797.
[50] E. Masazade, M. Fardad, and P. K. Varshney, "Sparsity-promoting extended kalman filtering for target tracking in wireless sensor networks," *(IEEE SPL)*, vol. 19, no. 12, pp. 845–848, 2012.
[51] D. J. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.
[52] P. Whaite and F. P. Ferrie, "Autonomous exploration: Driven by uncertainty," *(IEEE TPAMI)*, vol. 19, no. 3, pp. 193–205, 1997.
[53] R. Sim, "Stable exploration for bearings-only slam," in *(ICRA)*. IEEE, 2005, pp. 2411–2416.
[54] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[55] X. Wang, B. Golden, and E. Wasil, "A Steiner zone variable neighborhood search heuristic for the close-enough traveling salesman problem," *(COR)*, vol. 101, pp. 200–219, 2019.
[56] P. Delamatar, A. Finley, and C. Babcock, "Downloading and processing noaa hourly weather station data," *dim (st)*, vol. 1, no. 30538, p. 12, 2013.

Esmaeil Seraj is with the Institute for Robotics and Intelligent Machines (IRIM) at Georgia Tech, where he is working towards the Ph.D. degree in ECE. Prior to his PhD student, he was a visiting researcher at Georgia Tech and Emory University (2017-2019). Esmaeil received a MSc in ECE in 2016 from Shiraz University. His research lies in the intersection of controls and robot learning, collaborative multi-agent reinforcement learning, cooperative teaming and communication learning. Esmaeil is a finalist for the best student paper award at ACC 2020 and recipient of thesis support grant and publication awards by the COGC of Iran (2015-2017).

Letian Chen was born in Ningbo, China, in 1996. He received the Bachelor of Science in Psychology and Computer Science in 2018 from Peking University, Beijing, China. He received the Master of Science degree in Computer Science in 2020 from Georgia Institute of Technology, Atlanta, United States. He will join the School of Interactive Computing (IC) at Georgia Institute of Technology as a Ph.D. student in Computer Science starting 2020. His current research interests include Learning from Demonstration, Reinforcement Learning, Real-World Feasible Robot Learning, and Human Decision Making Modeling.



Matthew C. Gombolay is an Assistant Professor of Interactive Computing at the Georgia Institute of Technology. He received a B.S. in Mechanical Engineering from Johns Hopkins University in 2011, a S.M. in AeroAstro from MIT in 2013, and a Ph.D. in Autonomous Systems from MIT in 2017. Before starting as faculty, Dr. Gombolay worked at MIT's Lincoln Laboratory, transitioning his research to the U.S. Navy and earning an R & D 100 Award. His publication record includes an AIAA best paper award, an ACC best student paper nomination, and a CoRL best paper nomination. Dr. Gombolay is a DARPA Riser and recipient of a NASA Early Career Fellowship.