Received June 14, 2021, accepted July 5, 2021, date of publication July 12, 2021, date of current version July 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096229

Communication-Aware Multi-Agent Metareasoning for Decentralized Task Allocation

ESTEFANY CARRILLO[1], SUYASH YEOTIKAR[1], SHARAN NAYAK[1], (Graduate Student Member, IEEE), MOHAMED KHALID M. JAFFAR[1], SHAPOUR AZARM[2], JEFFREY W. HERRMANN[2], MICHAEL OTTE[1], (Member, IEEE), AND HUAN XU[1], (Member, IEEE)

[1]Department of Aerospace Engineering, University of Maryland at College Park, College Park, MD 20742, USA

[2]Department of Mechanical Engineering, University of Maryland at College Park, College Park, MD 20742, USA

Corresponding author: Estefany Carrillo (ecarril2@terpmail.umd.edu)

This work was supported by the Air Force Research Laboratory (AFRL) under Grant FA8750-18-2-0114.

ABSTRACT *Metareasoning* refers to reasoning about one's own decision making process. This paper considers metareasoning about the decision making process in multi-agent settings. We present a multi-agent metareasoning approach that enables a multi-agent team to select which task allocation algorithm to use as a function of changing communication quality level. Given a set of multi-agent task allocation algorithms, we synthesize a policy that prescribes the best algorithm to use among a predefined set of algorithms for a given communication level. Since each agent in the team runs the same policy, the team (or a part of the team) will collectively switch between task allocation algorithms as a function of the observed level of communication. We apply reactive synthesis to generate the policy from high-level specifications written in Linear Temporal Logic encoding the agents' switching behavior with respect to the state of the environment. We perform experiments in simulation to identify the best performing algorithms under different communication levels. The communication environment is modeled using the Rayleigh fading model and communication estimation is done through the exchange of heartbeat messages among agents. We test our metareasoning policy in three types of scenarios: search & rescue, fire monitoring, and ship protection scenarios. For each scenario, we demonstrate that our policy achieved better performance with respect to either max distance traveled, max number of transmitted messages or both compared to running any single algorithm.

INDEX TERMS Distributed robot systems, decentralized task allocation, meta-level control.

I. INTRODUCTION

Many of the benefits of Multi-Agent Systems (MAS) come from their ability to solve tasks more efficiently through collaboration. Some of the key challenges in MAS, such as the coordination of agents' behavior and distributed task allocation, require communication between agents. However, communication is unreliable in realistic environments and outside the control of the multi-robot team, making the coordination of tasks more challenging. Otte *et al.* [1] and Nayak *et al.* [2] have shown that different distributed task allocation algorithms perform better, relative to each other, at different communication quality levels.

To attain robustness to changes in communication, a naive strategy is to have each robot in the team use the best performing algorithm for the perceived level of communication. However, this may cause composability problems. For example, if communication quality varies over the environment, then different team members may select incompatible task allocation algorithms. If communication levels change over time, then switching between different task allocations algorithms may introduce additional overhead, create inefficiencies, or require restarting the task allocation process entirely. This raises questions on whether there are benefits to be gained from switching between different task allocation algorithms and what those benefits are.

The main motivation of this paper is to present a metareasoning framework that addresses these composability

The associate editor coordinating the review of this manuscript and approving it for publication was Rashid Mehmood.

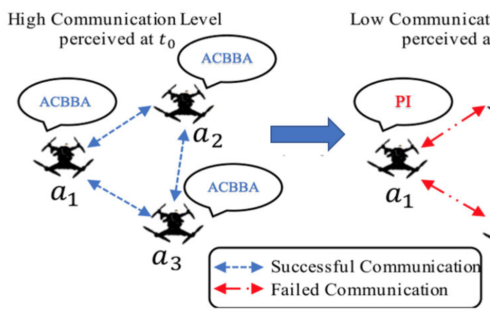


FIGURE 1. **A sample switching prescribed by the metareasoning policy for high and low communication levels. On the left, agents a_1, a_2 and a_3 perceive high communication availability at time t_0 and execute ACBBA as their task allocation scheme. On the right, agents switch to performing PI at time t_1 as a result of perceiving low communication availability.**

challenges and provides insights on these questions. *Metareasoning* describes reasoning about one's own decision making process. *Multi-agent Metareasoning* describes a process whereby a team of agents collectively reasons about the team's decision making process.

We propose a metareasoning approach that observes the communication quality in the environment and the state of each agent, individually, to enable a distributed team of agents to collectively switch between different task allocation algorithms without the need for synchronization. We synthesize a distributed multi-agent metareasoning policy (Fig. 1) from specifications written in Linear Temporal Logic (LTL). The synthesis is performed offline *a priori*, and the resulting policy is executed continually by each agent in real-time, adapting the team's distributed task assignment scheme in response to perceived changes in communication availability.

The novelty of our work is a meta-level controller that runs in parallel across the team, to improve system performance under changing communication conditions, when solving the problem of decentralized task allocation. Although meta-level control for the purpose of coordinating agents' behavior has appeared in [8], [9] and [13], our work is fundamentally different because we consider reasoning about a set of existing task allocation methods in order to make the system more robust to varying communication. This additional reasoning can lead to improved task allocation assignments even when communication between the agents deteriorates.

Our contributions are as follows:

1) We propose a metareasoning policy that enforces the team's prescribed behavior for task allocation as a function of the observed communication quality. In contrast to other metareasoning approaches which try to dynamically assess the benefit of additional reasoning at the cost of computational complexity, our policy is trained completely offline and synthesized from rule-based reasoning (LTL), limiting the computational effort required for online execution of the policy.
2) We demonstrate that the proposed multi-agent metareasoning approach achieves enhanced performance (i.e. lower communication and travel costs) when solving the problem of decentralized task allocation compared to running a single task allocation algorithm via an implementation.
3) We validate the proposed approach via a large number of experiments across multiple types of scenarios.

In this paper, the multi-agent metareasoning policy is built from a set of decentralized task allocation methods for which performance profiles under varying levels of communication were obtained in [2]. This set includes: the Consensus Based Auction Algorithm (**CBAA**) [14], the Asynchronous Consensus Based Bundle Algorithm (**ACBBA**) [15], [16], the Decentralized Hungarian Based Algorithm (**DHBA**) [17], the Hybrid Information and Plan Consensus (**HIPC**) algorithm [18], [19] and the Performance Impact (**PI**) algorithm [20]. To test the proposed metareasoning policy, we consider scenarios involving modern day operations such as surveillance, target classification and detection, search and rescue and ship protection operations, of particular interest to defense applications. Instantaneous communication conditions are modeled using a Rayleigh Fading model [21]. Changes in signal attenuation (i.e., communication quality) over time are modeled by varying the path loss exponent parameter. We adopt the Rayleigh fading model to simulate imperfect communication environments as it comes closer to modeling realistic environments compared to other models such as the Bernoulli or Gilbert-Elliot model models [2].

In addition, communication changes are triggered to happen at every channel in the environment. In each scenario, high or low communication are simulated for some period after which, the communication level is switched to low or high, respectively. We consider a fullmesh communication topology, a fully connected network, in which every agent *attempts* to communicate with every other agent and then messages are dropped according to the communication model. This assumption facilitates the discovery of performance differences due to applying the proposed policy as opposed to changing the agents' network topology [5]–[7]. For communication estimation, we propose a straightforward method in which each agent computes an estimate per communication link over which heartbeat messages are expected to arrive from every other agent in the environment. This method is well suited to provide link quality estimates for any underlying communication model. Other sources have explored probabilistic estimation methods of link quality with respect to channel power at different locations [3], but such estimation is beyond the scope of this paper.

This paper is organized as follows: Section II presents preliminaries including a description of each of the task allocation algorithms considered and communication model. The problem formulation is presented in Section III and the metareasoning framework is presented in Section IV. Section V describes each type of scenario implemented. Section VI describes the experimental setup and results are presented in Section VII. Finally, related work is discussed in Section VIII and conclusions appear in Section IX.

II. PRELIMINARIES

This section presents the metareasoning concepts, task allocation algorithms and communication model.

A. METAREASONING

Alexander *et al.* [40] developed a framework for metareasoning in MAS, in which agents consider and select their local problem-solving actions while coordinating with all other agents. In this framework, each agent solves a Markov Decision Process (MDP), which generates a computation policy that the agent uses to choose the best computation based on the current state. The agent's metareasoning procedure must alternate between its local decision making and the coordinated decision making among the entire group to account for various "what if" coordination scenarios.

In this work, we assume variable communication availability over time, thus in contrast to [40], we opt for synthesizing a distributed multi-agent metareasoning policy offline. We follow the multi-layered agent model outlined in [40]. Within this model, three types of actions are considered: *meta-level* actions, *object-level* actions and *ground-level* actions, which we will refer to as *low-level* actions (to avoid confusion with the use of the term "ground-level" in UAV applications). These actions are defined as follows:

1) *Low-level actions*: These actions are performed by the agent to change its state in the environment. Examples of such actions include movement, communication and sensing.
2) *Object-level actions*: These actions correspond to computational processes that output the low-level action to be performed by the agent to achieve its goal.
3) *Meta-level actions*: These actions are used to analyze and improve the performance of object-level actions.

In our framework, low-level actions include communication and point-to-point navigation, performed by each agent to complete its task sequence assignment. Object-level actions correspond to computing task sequence assignments using a specific task allocation method and performing communication estimation. At the meta-level layer, each agent executes a switching protocol. This protocol specifies the appropriate task allocation method (object-level action) for the perceived level of communication. Our research goal is to determine whether adding the proposed meta-level control layer improves performance in environments with varying communication levels.

B. TASK ALLOCATION ALGORITHMS

This section describes the algorithms considered for the coordination of task sequence assignments. For a detailed explanation of these algorithms, see their respective references.

CBAA is as an auction-based approach in which each agent obtains a single-task assignment [14]. This algorithm consists of two phases: the *assignment* phase and the *consensus* phase. In the *assignment* phase, each agent computes its local bids for all incomplete tasks and assigns itself the task with the lowest bid. The agent updates its winning bids list and sends it to all other agents. During the *consensus* phase, each agent updates its bids list with the lowest bids received and each task is assigned to the agent with the lowest bid.

ACBBA is a multi-task assignment algorithm [15], built as an extension of CBBA [41] for agents communicating asynchronously. This method operates in two phases: the *assignment* phase and the *consensus* phase. In the *assignment* phase, each agent gets assigned a bundle of tasks. Once bundles are built, agents update and share their winning bids lists along with the winning time stamps. In the *consensus* phase, agents resolve any conflicts found on their task assignments and update their internal lists.

The PI approach is presented in [20]. Similar to CBBA, PI assigns multiple tasks to agents, however, it uses a different kind of bid evaluation, known as significance. This evaluation is used to assess the contribution of a task to the cost of the current task sequence assignment. There are two phases in this approach: a *task inclusion* phase and a *consensus and task removal* phase. During the *task inclusion* phase, the marginal significance of unassigned tasks is computed in order to update the task bundle and significance lists. During the *consensus and task removal* phase, each agent shares its significance list with all other agents and does consensus by removing tasks for which the agent has been outbid by another agent. ACBBA consensus rules are used for this phase since we consider an asynchronous system.

DHBA is a task allocation algorithm that assigns a single task to each agent based on a cost matrix [17]. This matrix keeps track of the cost of each task for each agent. There are two phases in this algorithm: the *assignment* phase and the *update* phase. In the *assignment* phase, the Hungarian algorithm [42] is run on the cost matrix to obtain the optimal task assignment. In the *update* phase, agents broadcast the cost matrix and update it according to the information exchanged.

HIPC is a multi-task assignment algorithm [20], built as an extension of CBBA. However, instead of generating task bundles in a greedy fashion, HIPC tries to solve the task assignment problem for all agents at once. This algorithm has two phases: *task allocation* phase and *consensus* phase. The *task allocation* phase consists of each agent running a full Task Allocation Algorithm (TAA) to generate its task bundle. We run a variation of the nearest neighbors algorithm [12] using the min-max objective in the TAA implementation. In the *consensus* phase, agents resolve any conflicts on task assignments using ACBBA consensus rules.

For all bundle algorithms, the current task sequence is reset for each agent whenever a new target is dynamically added or removed from the workspace.

The next section describes the communication model used in our implementation.

C. RAYLEIGH FADING MODEL

Environmental clutter, e.g., buildings, trees, etc., tends to scatter radio signals and degrade communication quality. The propagated signals experience different shifts in amplitude, frequency and phase. The Rayleigh fading model predicts

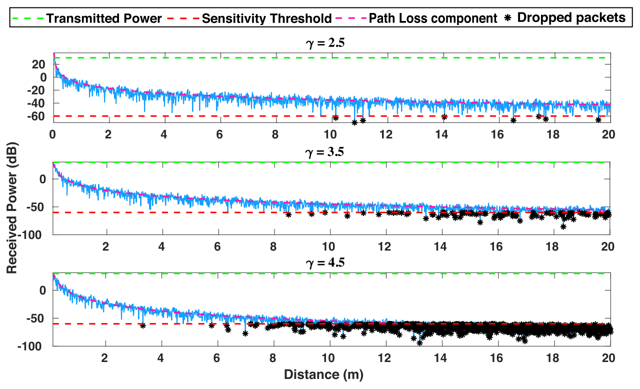


FIGURE 2. Received power of a signal attenuated by rayleigh fading and path loss is shown for three different values of the path loss exponent γ. The transmitted power $P_T = 30$ dB and the sensitivity threshold $P_S = -60$ dB. Asterisks '$*$' represent dropped packets. The number of dropped packets increases as the value of the path loss exponent increases.

the attenuation of the received signal by assuming that the signal's amplitude will vary according to a Rayleigh distribution [21]. This model has been widely used to simulate wireless communication in cases where it is desirable to account for both distance-based path loss effects as well as the random nature of dropped packets [21], [25] [26]. We note that alternative models exist that account for only random packet loss but not distance effects (Bernoulli model [1] and the Gilbert Elliot model [22], [23]) or only distance effects but not random path loss such as communication disk models [17]. Indeed, the Rayleigh fading model is widely used as model for wireless signal degradation; it is the special case of a Rician fading model [24] in which there exist sufficiently many signal propagation paths that the signal impulse response can be approximated as a Gaussian process. To quantify the fading effects, a Rayleigh random variate sequence is obtained efficiently using the Inverse Discrete Fourier Transform technique (IDFT) [44], [45]. We sample from the generated sequence, convert the sampled power value to decibel and calculate the attenuation due to fading P_F. In our implementation, in addition to fading effects, we also account for signal attenuation due to path losses. Eq. (1) is used to compute path losses P_{PL} as follows,

$$P_{PL} = P_{L_0} + 10\gamma \log_{10}\left(\frac{d}{d_0}\right), \tag{1}$$

where d is the current distance between the transmitter and receiver, γ is the path loss exponent and P_{L_0} is the path loss at reference distance d_0. The total attenuation is given by $P_L = P_F + P_{PL}$. The total received power is given by $P_R = P_T - P_L$, where P_T is the transmitted power. We define P_S as the user-specified sensitivity threshold. A message is dropped if the condition $P_R < P_S$ is satisfied as shown in Fig. 2 for three different communication scenarios. In each scenario, we show different amounts of dropped messages by setting $\gamma = 2.5$ (high communication), $\gamma = 3.5$ (medium communication) and $\gamma = 4.5$ (low communication).

III. PROBLEM FORMULATION

In this section, we present the metareasoning problem solved at the meta-level layer and the task allocation problem solved at the object-level layer.

A. METAREASONING PROBLEM

Consider a team of n agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ and the multi-agent task allocation problem, denoted as $\mathcal{P}$, that $\mathcal{A}$ needs to solve. Agents can choose from a set of l multi-agent task allocation algorithms, $A = \{A_1, \ldots, A_l\}$, to solve $\mathcal{P}$. For instance, we can set $\mathcal{P} = search\&rescue$ and $A_1 =$ CBAA. Let $\mathcal{E}$ denote the space of environmental features such as communication level and target density. Environmental features are allowed to change as functions of time. Thus, a realization of environmental features is denoted as $e(t) \in \mathcal{E}$. Though we do not have direct access to the true value of $e(t)$ at any instant of time t, we can obtain an estimate of $e(t)$, denoted as $\tilde{e}(t)$. Consider $\tilde{e}_{i;[0:k_i]} = \{\tilde{e}_{i;0}, \tilde{e}_{i;1}, \ldots, \tilde{e}_{i;k_i}\}$ to be the sequence of the first $k_i + 1$ estimates computed from agent a_i. Let $\mathcal{E}_n$ be the space of all possible sequences of estimates from a team of size n. Let M be the multi-agent metareasoning approach.

Definition 1 (Instantaneous Multi-Agent Metareasoning Problem): Given $\mathcal{P}$ and a sequence of observations from all n agents, $\{\tilde{e}_{1;[0:k_1]}, \ldots, \tilde{e}_{n;[0:k_n]}\} \in \mathcal{E}_n$, where $k_i + 1$ represents the number of observations obtained by the *i-th* agent, the instantaneous multi-agent metareasoning problem is to calculate the tuples $(A_i, T_i) = M(\{\tilde{e}_{1;[0:k_1]}, \ldots, \tilde{e}_{n;[0:k_n]}\})$ composed by the multi-agent algorithm $A_i \in A$ as well as the subteam $T_i \subseteq \mathcal{A}$ that will use algorithm A_i so that the team $\mathcal{A} = \dot{\bigcup}_{i=1}^{L} T_i$, where L is the number of tuples generated, communally solves $\mathcal{P}$. Note that the subteams are disjoint sets of $\mathcal{A}$ since agents cannot belong to multiple subteams simultaneously.

Definition 2 (General Multi-Agent Metareasoning Problem): For time steps $1, \ldots, t$, repeatedly solve the Instantaneous Multi-Agent Metareasoning Problem, and then have agents in each subteam T_i use, respectively, multi-agent algorithm A_i to solve $\mathcal{P}$.

B. DECENTRALIZED TASK ALLOCATION PROBLEM

The problem of decentralized task allocation can be formulated as a binary integer programming problem, similar to the multiple Traveling Salesman Problem (mTSP) [15]. Given a set of m tasks and a set of n agents, a solution is obtained such that each agent is assigned a sequence of tasks and every task in the sequence is completed by the agent. Consider a set of m tasks $\mathcal{T}$. Let $S_i \subseteq \mathcal{T}$ be a sequence of tasks assigned to agent a_i and p_i be the number of tasks in S_i. Let $q(S_i)$ be the cost of sequence S_i. The goal of the decentralized task allocation problem is to obtain a sequence S_i for each agent $a_i \in \mathcal{A}$ such that these sequences are disjoint and $\mathcal{T} = \bigcup_{i=1}^{n} S_i$.

The next section describes the metareasoning framework in which the metareasoning and the task allocation problems are solved.

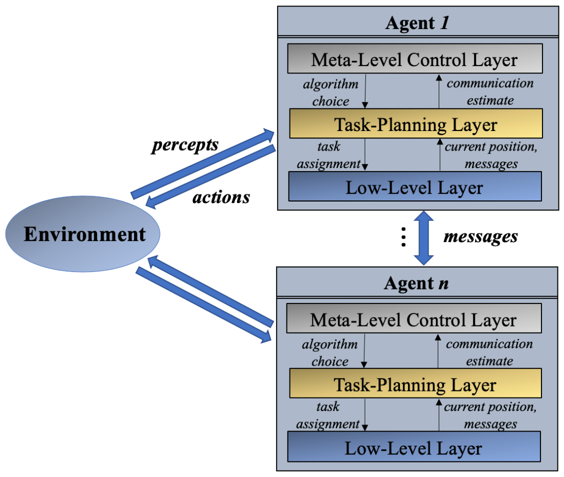


FIGURE 3. **Control flow within the metareasoning framework proposed. Each agent has a meta-level control layer, a task-planning layer and a low-level layer. Agents compute a communication estimate from the messages received at the low-level layer. Using this estimate, the meta-level control layer outputs the algorithm choice for the agent.**

IV. METAREASONING FRAMEWORK

Our metareasoning framework defines three control layers (shown in Fig. 3) in every agent:

(i) *Meta-level layer*: This layer decides on the decentralized task allocation algorithm the agent should perform (meta-level action).
(ii) *Task-planning layer*: This layer decides on the task sequence assignment (object-level action).
(iii) *Low-level layer*: This layer generates the trajectories along which the agent needs to move to reach each task location.

Each agent's decision cycle consists of:

(a) Estimating the communication level in the environment at the task-planning layer from heart-beat messages received at the low-level layer.
(b) Executing meta-level control to output the appropriate algorithm for the perceived communication level according to the metareasoning policy.
(c) Performing the chosen algorithm to obtain a task sequence assignment.

The novelty of our approach lies in the policy placed at the meta-level control layer. This layer consists of a switching protocol as defined by a fixed, common metareasoning policy computed offline. This ensures that agents perform the same algorithm for a given level of communication without the need to communicate or synchronize their decisions during runtime. Since agents compute communication quality estimates locally and run an independent copy of the policy, different agents may have different beliefs about the communication quality at any instant of time and therefore may use different algorithms simultaneously.

In our implementation, we frame the synthesis of the metareasoning policy as a reactive synthesis problem, which can be solved by digital design synthesis tools [46].

TABLE 1. **LTL notation.**

LTL Symbol	Definition
$\Box$	Always
$\Box\Diamond$	Always Eventually
$\bigcirc$	Next

The synthesis problem can be viewed as a two-player game between an environment that attempts to falsify the specification and the system that tries to satisfy it. Thus, we synthesize the metareasoning policy as the solution to the reactive synthesis problem involving the communication level in the environment and the agent's algorithm choice. We note that simpler methods can be used to implement the policy such as a fuzzy logic controller or adhoc heuristics, however, the formal framework will enable us to easily encode different types of high-level specifications involving behaviors such as sequencing, conditions and avoidance, in future work.

In this work, specifications involve temporal logic, which is an extension of Boolean logic with temporal semantics. The building blocks of LTL formulas consist of Boolean variables, logical connectives and temporal modal operators. The logical connectives include: negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and material implication ($\longrightarrow$). The temporal modal operators include next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$). For a more detailed discussion of LTL, see the preliminaries section of [46]. One of the most commonly used forms for these LTL formulas is the General Reactivity(1) (GR(1)) assume-guarantee form [46], shown in Eqs. (2)-(4),

$$\varphi_e \rightarrow \varphi_s, \tag{2}$$
$$\varphi_e = \varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e, \tag{3}$$
$$\varphi_s = \varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s, \tag{4}$$

where φ_e characterizes the assumptions on the environment and φ_s characterizes the system requirements, φ_i^e, φ_i^s are the initial values for the environment and system variables, φ_t^e, φ_t^s represent the evolution of the state of the environment and system, and φ_g^e, φ_g^s represent goal assumptions for the environment and desired goal specifications for the system.

The LTL specifications in GR(1) form are used to solve the synthesis problem in polynomial time [46]. For these specifications, the synthesizer in the temporal logic planning (TuLiP) toolbox [43] can derive a finite state machine in which states are valuations of environment and system variables and transitions correspond to actions that the system can take to reach a desired state. We summarize the main LTL notation used in this Section in Table 1.

The following sections define the assumptions on the environment and the system requirements.

A. ENVIRONMENT

Each agent sends a fixed number of heartbeat messages periodically for the purpose of communication estimation. These heartbeat messages do not contain any meaningful information, but their absence can signal loss of communication between a sender and a receiver. Each agent a_i estimates the per link communication with agent a_j by computing the ratio of the number of heartbeat messages received per link h_{ij} and the expected number of heartbeat messages h. At each time step, agent a_i can determine its communication estimate c_{e_i} by computing the max ratio over all its communication links,

$$c_{e_i} = \max_{a_i \neq a_j, a_j \in \mathcal{A}} (\frac{h_{ij}}{h}). \tag{5}$$

Eq. (5) captures the overall communication change in the environment. For instance, as communication degrades, the max ratio across all channels will decrease, assuming that communication drops across all channels of communication. Similarly, as communication improves, the max ratio will capture the increase in communication availability. Thus, we consider this estimation method to be suitable as changes in communication are triggered across every communication channel. Such communication changes can be encountered in scenarios with changing weather conditions or a changing presence of dynamic obstacles including adversarial agents that seek to interfere with the communication among agents.

The communication estimate c_{e_i} is then mapped to a discrete communication level. Nayak *et al.* [2] showed that the performance ranking of the task allocation algorithms tested remains the same at high communication levels for the visit and search & visit scenarios. It is only when communication drops substantially that this ranking changes and remains constant for lower levels of communication. For this reason, we define only two discrete communication levels based on the analysis presented in [2]: high (H) for $c_t < c_{e_i} \leq 1$ and low (L) for $0 \leq c_{e_i} \leq c_t$. The threshold value c_t is specified experimentally by mapping the sensitivity threshold values tested in [2] to either high or low communication based on the sensitivity threshold value at which the change in ranking of the algorithms was observed. Details on these experiments are presented in Section VI-B.

In the synthesis problem, each of these discrete communication levels is represented by a Boolean variable. Thus, the environment in our problem can be described by these two variables as $E = H \times L$. This implies that the state of the environment corresponds to a valuation of these two Boolean variables. If the communication estimate c_{e_i} is within the range $c_t < c_{e_i} \leq 1$, then H will be set to *True* and L will be set to *False*. Otherwise, if $0 \leq c_{e_i} \leq c_t$, H will be set to *False* and L will be set to *True*. The environment specifications are as follows:

$$\varphi_i^e = L \wedge \neg H, \tag{6}$$
$$\varphi_t^e = \square(\neg(H \wedge L) \wedge (H \vee L)), \tag{7}$$
$$\varphi_g^e = \square\Diamond H \wedge \square\Diamond L. \tag{8}$$

Eq. (6) states that initially the environment is assumed to have a low communication level. This assumption is valid if we expect it will take a few milliseconds for all agents to start up the exchange of messages. Though, any initial condition can be chosen as long as it satisfies all other specifications.

Eq. (7) is required to verify that exactly one of the two variables H and L is true at all times. This is specified to ensure that the environment is in a single level of communication, ignoring cases in which communication is simultaneously high and low or simultaneously neither. Finally, Eq. (8) states that always eventually: the communication level of the environment will be high and that always eventually: the communication level of the environment will be low. This specification ensures that the communication in the environment will change at some future time.

B. SYSTEM

Let $S = \{A_j, A_k\} \times \{True, False\}$ represent the system states, where $\{A_j, A_k\} \subset A$. An element $s \in S$ represents the tuple composed by the multi-agent task allocation algorithm used by the agent and a Boolean variable, denoted as *Reset*. This variable will be set to *True* if the agent's current task sequence assignment and bids need to be reset before executing task allocation. If *Reset* is set to *False*, then the agent continuous on using its current task sequence assignment and winning bids information. The rationale for this is that as communication improves, agents are likely to obtain better solutions and may benefit from discarding lower quality solutions obtained under poor communication.

Let $F \subseteq S \times S$ be the set of all possible transitions between the elements in the state space. When switching happens, agents only need to transfer their current winning bids, completed tasks and current task sequence assignment into the execution of an iteration of the new algorithm. We define the following system specifications:

$$\varphi_i^s = A_k \wedge Reset, \tag{9}$$
$$\varphi_{t,1}^s = \square((L \wedge \bigcirc H) \rightarrow \bigcirc(A_j \wedge Reset)), \tag{10}$$
$$\varphi_{t,2}^s = \square((H \wedge \bigcirc L) \rightarrow \bigcirc(A_k \wedge \neg Reset)), \tag{11}$$
$$\varphi_{t,3}^s = \square((L \wedge \bigcirc L) \rightarrow \bigcirc(A_k \wedge \neg Reset)), \tag{12}$$
$$\varphi_{t,4}^s = \square((H \wedge \bigcirc H) \rightarrow \bigcirc(A_j \wedge \neg Reset)). \tag{13}$$

Eq. (9) states that agents reset their task sequence assignments and perform A_k initially. Eqs. (10)-(13) indicate which algorithm the agent should perform and whether or not the agent should reset its current task sequence assignment next based on the current and next states of the environment. Eq. (10) specifies that, when communication improves, agents should reset their task sequence assignments. Eqs. (11)-(13) specify that, when communication degrades or remains the same, agents should not reset the bids information.

For each scenario, the best performing algorithms are identified for high and low communication levels, respectively, via simulation experiments. We set A_j to the best performing algorithm under high communication and A_k to the best performing algorithm under low communication. For the search & rescue scenario, we set A_j = ACBBA and A_k = CBAA by leveraging the results from [2]. Further experiments are performed to identify A_j and A_k for the fire monitoring and ship protection scenarios. These experiments will be described in Appendix D.

C. ALTERNATIVE TEAM UP STRATEGY

If agents perceive the same communication quality level, they will execute the same task allocation algorithm specified for that level. However, when agents switch to performing different algorithms simultaneously because of having different beliefs about the communication quality in the environment, they may compute task assignments from outdated information. This information may have been successfully received at some point in time, but may no longer be valid once the agents lose communication and change their task allocation schemes. To address some potential conflicts from changes in communication, we propose a *team up* strategy in which agents fully coordinate only with agents from which they perceive a high level of communication. In addition to switching to the appropriate task allocation algorithm for a given communication scenario, each agent applies this *team up* strategy, in which, if the agent *teams up* with another agent, it will use all the bids information received from the agent when executing task allocation. On the contrary, if the agent perceives low communication from another agent, it will discard all bids information received from this agent except for information pertaining completed tasks. Fig. 4 shows agents a_0, a_1 and a_2 performing the *team up* strategy.

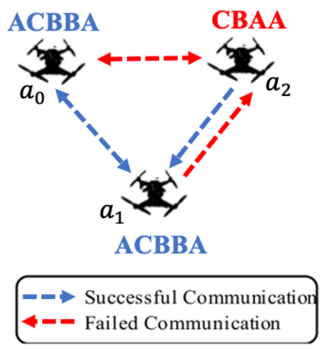


FIGURE 4. Sample execution of the synthesized switching protocol where agents team up according to the level of communication perceived. Agents a_0 and a_1 team up as they perform ACBBA, while a_2 individually performs CBAA.

The motivation for this strategy is to prevent agents from relying on all the information received from agents with which they experience poor communication. Consider an agent a_1 that receives messages about tasks that another agent a_2 intends to do and plans accordingly. However, if a_2 decides to perform different tasks and broadcasts messages about its updated plan, these messages may not be received by a_1. This could lead to a_1 and a_2 leaving some tasks incomplete or taking much longer to complete all tasks. By having agents select which information to include in their computations based on communication availability, tasks can be completed even when broadcasted messages are not received.

TABLE 2. Task allocation problem, task definition and objective function type for each scenario type. Scenarios are ordered in increasing level of difficulty.

Task Allocation Problem	Task Definition	Objective Function
Search & Rescue	Known grid cells and unknown stationary targets	$\min_{\mathcal{X}} \left(\max_{a_i \in \mathcal{A}} q(S_i)\right)$
Fire Monitoring	Known grid cells and unknown fire targets spreading	$\min_{\mathcal{X}} \left(\max_{a_i \in \mathcal{A}} q(S_i)\right)$
Ship Protection	Known grid cells and unknown moving targets	$\max_{\mathcal{X}}(-k_0 F(\mathcal{X}) + k_1 h_1(\mathcal{X}) + k_2 h_2(\mathcal{X}))$

The following section describes the decentralized task allocation problem $\mathcal{P}$ considered for each scenario implemented given a team of agents $\mathcal{A}$.

V. DECENTRALIZED TASK ALLOCATION SCENARIOS

The scenarios considered for our experiments are summarized in Table 2. and simulation runs for each type of scenario are shown in Fig. 5.

We define $\mathcal{T}$ and $q(S_i)$ for each type of scenario as follows:

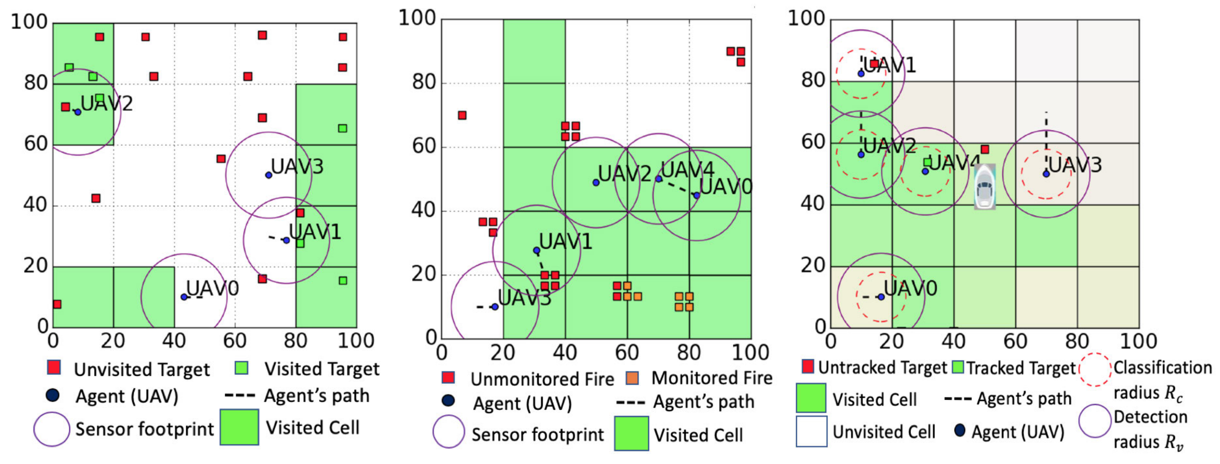
1) SEARCH & RESCUE SCENARIO

In this scenario, we define $\mathcal{T} \triangleq \mathcal{G} \cup \mathcal{U}$, where $\mathcal{G} = \{g_1, \ldots, g_r\} \subset \mathcal{W}$ is a finite set of *a priori* known grid cells and $\mathcal{U} = \{u_1, \ldots, u_m\} \subset \mathcal{W}$ is a finite set of unknown stationary targets located in a map, $\mathcal{W} \subset \mathbb{R}^2$, of size $N \times N$. The cells in $\mathcal{G}$ divide the search space into regions of equal size. Initially, agents begin searching the map by visiting each cell in $\mathcal{G}$. Each cell is said to be completely searched when an agent reaches its center because, at this location, an agent's sensor radius R_d covers the entire cell, and the agent can detect any targets in that cell.

As agents search the space, they are able to detect new targets located within their sensor radius. Discovered targets are added to the set of known tasks $\mathcal{K}$, which is equal to $\mathcal{G}$ at the start of the mission. Agents share information about the newly discovered targets with other agents. Thus, S_i is the sequence of p_i tasks (stationary targets and grid cells) in $\mathcal{K}$ that are assigned to a_i. A target is considered to be visited when an agent moves within a threshold distance δ_T of the target's location. The mission is completed when every cell is searched by at least one agent and every target is visited by at least one agent.

The *min-max* objective $\mathcal{O}(\mathcal{X})$ considered in this scenario is to find a task assignment $\mathcal{X}^* = \{S_1, \ldots, S_n\}$ such that

$$\mathcal{O}(\mathcal{X}^*) = \min_{\mathcal{X}} \left(\max_{a_i \in \mathcal{A}} q(S_i) \right). \qquad (14)$$



(a) Search & rescue scenario. (b) Fire monitoring scenario. (c) Ship protection scenario.

FIGURE 5. Possible runs of agents performing decentralized TA for each type of scenario considered in our problem formulation. Each target has a unique ID, not shown in these figures to reduce clutter. In the ship protection scenario, lighter shades of green reflect that a longer time has passed since cells were last searched.

The cost function $q(S_i)$ is defined as

$$q(S_i) = C_i + c_i(u_1) + \sum_{k=1}^{p_i-1} \|u_{k+1} - u_k\|, \qquad (15)$$

where C_i is the cost accrued by a_i up to its current location and $c_i(u_k)$ corresponds to a_i's cost of visiting target u_k. The cost $c_i(u_k)$ is calculated as the Euclidean distance from the agent a_i to u_k. Thus, the min-max objective is to minimize the maximum distance traveled over all agents. Assuming a constant speed for the agents, this is equivalent to minimizing the mission completion time.

2) FIRE MONITORING

We define $\mathcal{T} \triangleq \mathcal{G} \cup \mathcal{U}$ as in the search & rescue scenario with the difference that stationary targets in this scenario correspond to fire locations which are dynamically generated and added to the map during the mission. Fire locations are propagated according to the wavelet differential equations and the Rothermel spread equation provided in Farsite [32], a fire simulation engine. At each simulation step, fire grows continuously for a time interval Δt, after which new fire locations are discretized into fire regions. Each fire region obtained is considered as an stationary target. Fire locations are propagated until a max number of targets is reached. Discovered fire targets are added to the set of known tasks $\mathcal{K}$, which is equal to $\mathcal{G}$ initially. Agents share locations of the newly discovered fire targets with other agents. Thus, S_i is the sequence of p_i tasks (fire targets and grid cells) in $\mathcal{K}$ that are assigned to a_i. The mission is completed when every cell has been searched by at least one agent and every fire target has been visited by at least one agent.

The *min-max* objective $\mathcal{O}(\mathcal{X})$ and cost function $q(S_i)$ in this scenario are defined in the same way as in the search & rescue scenario.

3) SHIP PROTECTION

In this scenario, we consider a ship with location (s_x, s_y), initially placed at the boundary of the map. During the mission, the ship moves with a constant speed v_s and constant heading θ_s to the opposite side of the map. The tasks in this scenario correspond to the set $\mathcal{T} \triangleq \mathcal{G} \cup \mathcal{U}$, where $\mathcal{G} = \{g_1, \ldots, g_r\} \subset \mathcal{W}$ is a finite set of *a priori* known grid cells and $\mathcal{U} = \{u_1, \ldots, u_n\} \subset \mathcal{W}$ is a finite set of unknown moving targets. Unlike all previous scenarios, cells here are searched continuously since targets can move across different cells and might appear in a cell already searched. Each agent knows, for each cell g_i, the elapsed time t_{g_i} since the last time that some agent searched that cell. A cell is said to be searched when an agent reaches the center of the cell, at which point t_{g_i} is reset to 0. Agents broadcast the locations of all newly discovered targets in the cells and the timestamps at which the cells were last searched.

As agents search the space, they can detect, classify and track, if necessary, all moving targets in the map in order to protect the ship. There are two types of moving targets in the set $\mathcal{U}$: adversarial and non-adversarial. Adversarial targets move towards the ship using a directed random walk while non-adversarial targets move randomly in the map. All targets move along piece-wise linear trajectories and remain within a threshold distance δ_s from the ship. Let θ_{u_i} be the heading of each adversarial target $u_i \in \mathcal{U}$. This heading is changed randomly at fixed intervals of time, otherwise it is computed using the current location of the ship (s_x, s_y) and the location of the target (u_{ix}, u_{iy}) as

$$\theta_{u_i} = \tan^{-1}\left(\frac{s_y - u_{iy}}{s_x - u_{ix}}\right). \qquad (16)$$

The heading for each target is sampled from $[-90°, 90°]$ and the target speed can vary between 0 and v_u, which is set to be strictly less than the agents' max speed. We do this to ensure that the agents succeed in tracking all the targets.

In addition to the sensor radius R_d that each agent uses for detecting targets, we define a classification radius R_c. When a target moves within an agent's classification radius, it can be classified as adversarial or non-adversarial by the agent. We set $R_c < R_d$ in order to move agents closer to targets in their effort to classify them. If the target is found to be adversarial, the agent proceeds to track it by moving towards the target using a proportional controller with control law,

$$\boldsymbol{w} = k_p(\boldsymbol{x_{ui}} - \boldsymbol{x_{ai}}) + \boldsymbol{v_{ui}}, \tag{17}$$

where k_p is the controller gain, $\boldsymbol{x_{ui}}$ is the current position of target u_i, $\boldsymbol{x_{ai}}$ is the current position of agent a_i, and $\boldsymbol{v_{ui}}$ is the current velocity of u_i. An adversarial target is considered to be tracked when an agent moves within a threshold distance δ_T of the target's location. A tracked adversarial target will move away from the ship towards the boundary of the map and eventually leave the map for the rest of the mission. A non-adversarial target successfully classified by an agent is considered to be tracked immediately after. A tracked non-adversarial target will continue to move along its trajectory unaffected by the agent's actions.

Newly discovered targets are added to the set of known tasks $\mathcal{K}$, initially set equal to $\mathcal{G}$. Newly tracked targets are added to the set $\mathcal{Z} \subseteq \mathcal{U}$, which is empty at the start of the mission. Thus, S_i is the sequence of p_i tasks (grid cells and moving targets) in $\mathcal{K}$ that are assigned to a_i. The mission is finished when the ship reaches the opposite side of the map.

The *max* objective $\mathcal{O}(\mathcal{X})$ considered in this scenario is to find a task assignment $\mathcal{X}^* = \{S_1, \ldots, S_n\}$ such that

$$\mathcal{O}(\mathcal{X}^*) = \max_{\mathcal{X}} -k_0F(\mathcal{X}) + k_1h_1(\mathcal{X}) + k_2h_2(\mathcal{X}), \tag{18}$$

where $F(\mathcal{X})$ is the max travel cost to be accrued by each agent $a_i \in \mathcal{A}$ performing its assigned task sequence S_i over all agents. This term is weighted by k_0, a user-specified value. We define this term as

$$F(\mathcal{X}) = \max_{a_i \in \mathcal{A}} q(S_i). \tag{19}$$

The cost function $q(S_i)$ is defined as in Eq. (15) where $c_i(u_k)$ corresponds to agent a_i's cost of tracking a moving target or visiting a grid cell in $\mathcal{K}$.

The second term in the objective is the expected distance to the ship over all unassigned cells and is defined as

$$h_1(\mathcal{X}) = \min_{g \in \mathcal{G} \setminus \bigcup_{a_i \in \mathcal{A}} S_i} p^{t_g} d_s(g), \tag{20}$$

where p is a value in (0, 1), t_g is the time elapsed since cell g was searched and $d_s(g)$ is the distance between the center of the cell and the location of the ship. The effect of $h_1(\mathcal{X})$ is to prioritize searching unassigned cells that are closer to the ship and that have not been visited for a longer time.

The third term in the objective is the minimal distance to the ship over all untracked targets and is defined as

$$h_2(\mathcal{X}) = \min_{u \in \mathcal{K} \setminus \mathcal{Z}} d_s(u), \tag{21}$$

where $d_s(u)$ is the distance between the target's location and the ship. The effect of $h_2(\mathcal{X})$ is to prioritize tracking targets that are closer to the ship. The terms $h_1(\mathcal{X})$ and $h_2(\mathcal{X})$ are weighted by k_1 and k_2, respectively, both of which are user-defined coefficients. Thus, the max objective is equivalent to a weighted combination of minimizing the maximum distance traveled over all agents, maximizing the minimum expected distance to the ship over all unassigned cells and maximizing the minimum distance to the ship over all untracked targets.

A second objective that we consider across all scenarios is the min-max number of transmitted messages, whether received or not. Both performance metrics, max travel distance (equivalent to mission time) and number of transmitted messages, are critical in a number of applications in which agents may experience limited communication such as area coverage [47], environmental monitoring [49], emergency management [50], and search & rescue missions [51].

VI. EXPERIMENTAL SETUP

The simulation framework is built in the Robot Operating System (ROS) [52] Kinetic. Two types of modules are implemented: agent and environment. The task allocation algorithms, written in Python, are executed in the processing unit of each agent module every 0.1s. Once a task assignment is obtained, the agent starts moving towards the task location unless it receives a message from another agent indicating that the task has already been completed. Only when the agent arrives at its assigned task will it proceed to complete its next assigned task. The agents communicate their solutions and heartbeat messages over the ROS network through a communication interface written in C++. Separate processes for all agents are used in order to simulate a decentralized system. Only one process is used for the environment module. The environment module simulates the agents as point robots as well as the 2D map in which the agents move. We assume a collision free model for the agents. To move in the map, the agent module sends a request to the environment simulator, which provides odometry sensor readings for the agent. For more details about this framework, see [2].

We ran all simulations on an AMD Ryzen Threadripper 2990WX, 32-core, 3 GHz CPU with 32 GB RAM. In all scenarios except for the ship protection scenario, simulations were terminated when at least one agent came to know all target locations had been visited. In the ship protection scenario, simulations were terminated when the ship arrives to the other side of the map.

Regarding the communication topology, we assume a *full mesh* topology in which every agent attempts to communicate with every other agent by continuously broadcasting messages. The Rayleigh Fading model is used to determine whether or not a message is dropped. Thus, the topology of the network is sparse and changes as a function of space and time.

At the beginning of each simulation, a list of winning bids is initialized to a large number of tasks for all agents. This list is set so that all bids are set to a very large value. When performing ACBBA or any other multiple-task assignment method, a sequence of n tasks is derived from the list of winning bids. In the case of CBAA or any other single-task assignment method, n is set to 1.

A. DETERMINATION OF COMMUNICATION THRESHOLD VALUE

Using the Rayleigh Fading model, results from [2] showed that the performance ranking of the task allocation algorithms tested changed at a sensitivity threshold value of -25 dB for the visit scenario and -35 dB for the search & visit scenario. Thus, we define values below -35 dB as high communication and values above -35 dB as low communication. For each sensitivity threshold value tested in [2], we ran experiments to compute the corresponding communication estimate. We then set the value of c_t to the communication estimate at which communication levels change from low to high. Theoretically, one could vary the sensitivity threshold values to simulate high and low communication as done in [2]. However, this would not be feasible in a physical implementation since the sensitivity threshold is inherent to the agent's hardware. Instead, in this work, we choose to vary the path loss exponent γ, which is equivalent to generating different amounts of clutter in the environment [53]. In addition, we choose to trigger communication changes at every channel in the environment. This would apply to scenarios in which the environment is changing at a large scale due to, for example, changing weather conditions. We consider this to be a first step for simulating and testing communication changes in the environment which can happen in more complicated ways. For instance, select regions of the environment can experience more clutter than other regions which would require triggering communication shifts per communication channel. However, simulating such communication shifts is beyond the scope of this work. We describe the experiments ran to set the value of c_t and the ranges of γ values in Appendix A.

B. DESIGN OF METAREASONING EXPERIMENTS

To test the metareasoning policy, we first identified the best performing algorithms under high and low communication levels for each type of scenario. For the search & rescue scenario, we selected the best performing algorithms based on the results from [2]. For the fire monitoring and ship protection scenarios, we ran experiments to characterize the performance of each of the 5 algorithms considered under different levels of communication. These experiments were performed using the optimal algorithm parameters for each scenario and the optimal weighting coefficients k_0, k_1 and k_2 in the objective function for the ship protection scenario. The experiments ran to determine k_0, k_1 and k_2 are described in Appendix B. Details on the experiments ran to find the optimal algorithm parameters for the fire monitoring and ship protection scenarios are presented in Appendix C.

TABLE 3. Algorithm specified by the policy for each level of communication and type of scenario.

Scenario Type	High Comm. Alg.	Low Comm. Alg.
Search & rescue	ACBBA	CBAA
Fire monitoring	ACBBA	CBAA
Ship protection	PI	DHBA

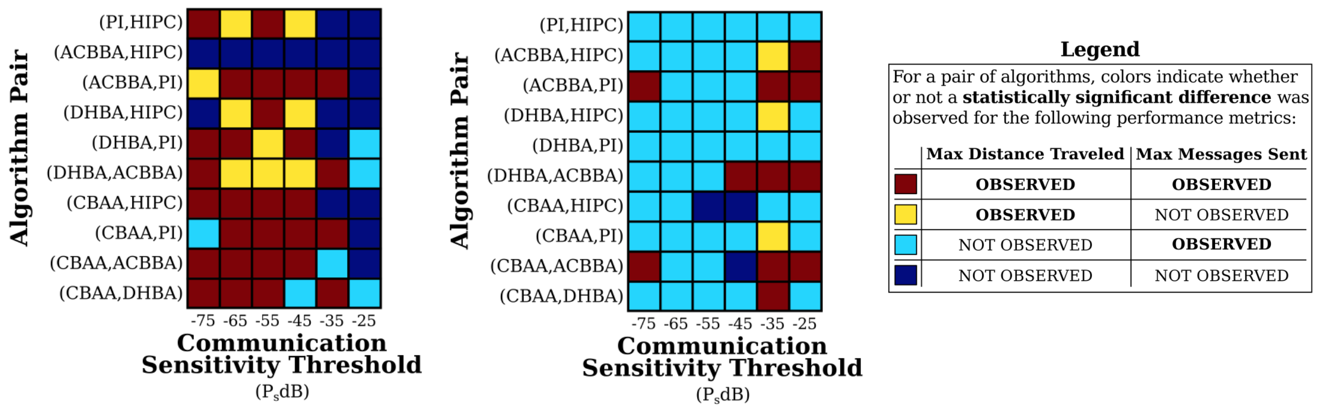
To evaluate our metareasoning policy, we tested various initial conditions for the number of agents and the number of tasks. A description of the experiments used to evaluate the performance of each algorithm as well as the metareasoning policy for all the scenario types considered can be found in Appendix D. Each scenario instance is generated by setting the number of agents and the number of targets in addition to other scenario specific parameters via uniform random sampling from specified ranges. A summary of these ranges is shown in Appendix D.

Our policy as well as each algorithm used in the policy were tested on multiple instances of each scenario type at two different switching conditions of communication: low to high and high to low. These conditions represent possible ways in which communication may change during a mission. We sampled values for γ from $[2.0, 3.0]$ to simulate high communication and from $[4.5, 5.0]$ to simulate low communication. We set the switching time for communication t_1 as a random value in $[5, 15]$. The heartbeat rate is set to 5 messages per second. A total number of 50 instances was generated for each type of scenario and communication switching condition. The total number of experiments was $50 \times 2 \times 4 = 400$.

C. SELECTION OF ALGORITHMS

We selected the algorithms for the metareasoning policy for each type of scenario from the results obtained. See Table 3. for a summary of these results. We show statistical analysis results obtained using the Wilcoxon Rank Test (WSR) [55] in Fig. 6 and Metrics Trade-off plots (MTP) in Fig. 7.

For the fire monitoring scenario, Fig. 7 shows that ACBBA, HIPC, DHBA and CBAA were the best performing algorithms on average at high communication, while ACBBA, CBAA and PI were the best performing algorithms on average at low communication. From the WSR test results, we observe that at high communication, algorithm pairs (CBAA, ACBBA) and (DHBA, ACBBA) exhibit statistical difference in messages transmitted and max distance traveled, while (ACBBA, HIPC) shows no statistical difference in either performance metric. Since the MTP shows that ACBBA obtains the minimum max distance traveled, we select ACBBA at high communication. At low communication, we note that (ACBBA, CBAA) shows no statistical difference in max distance traveled, while (PI, CBAA) shows no statistical difference in either metric. Thus, we choose CBAA over PI since it achieves the lowest number of messages transmitted on average as shown in the MTP.

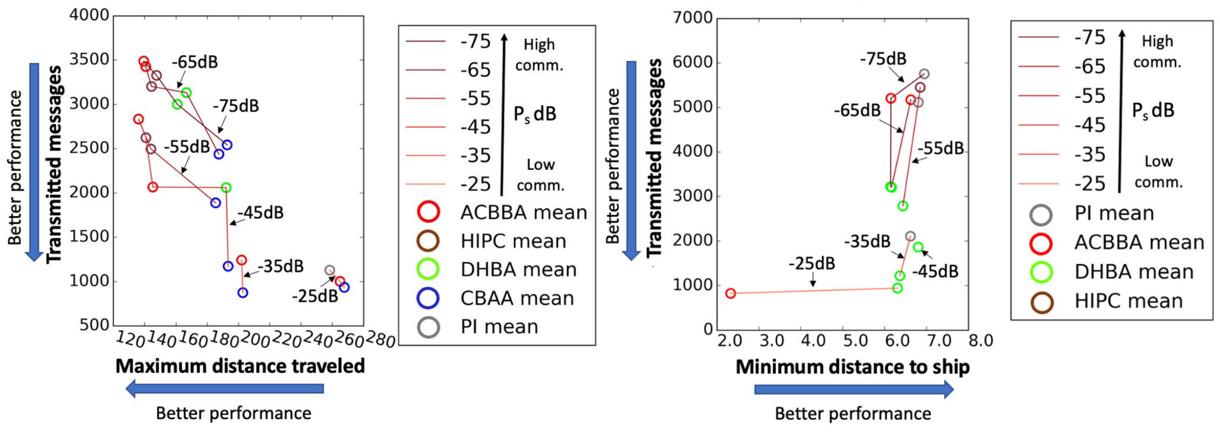


For a pair of algorithms, colors indicate whether or not a **statistically significant difference** was observed for the following performance metrics:

	Max Distance Traveled	Max Messages Sent
🟥	**OBSERVED**	**OBSERVED**
🟨	**OBSERVED**	NOT OBSERVED
🟦	NOT OBSERVED	**OBSERVED**
⬛	NOT OBSERVED	NOT OBSERVED

(a) Fire Monitoring Scenario. (b) Ship Protection Scenario.

FIGURE 6. For each communication level and pair of algorithms tested, we show the statistically significant differences observed using the Wilcoxon Signed Rank test for the fire monitoring and ship protection scenarios with respect to the performance metrics.



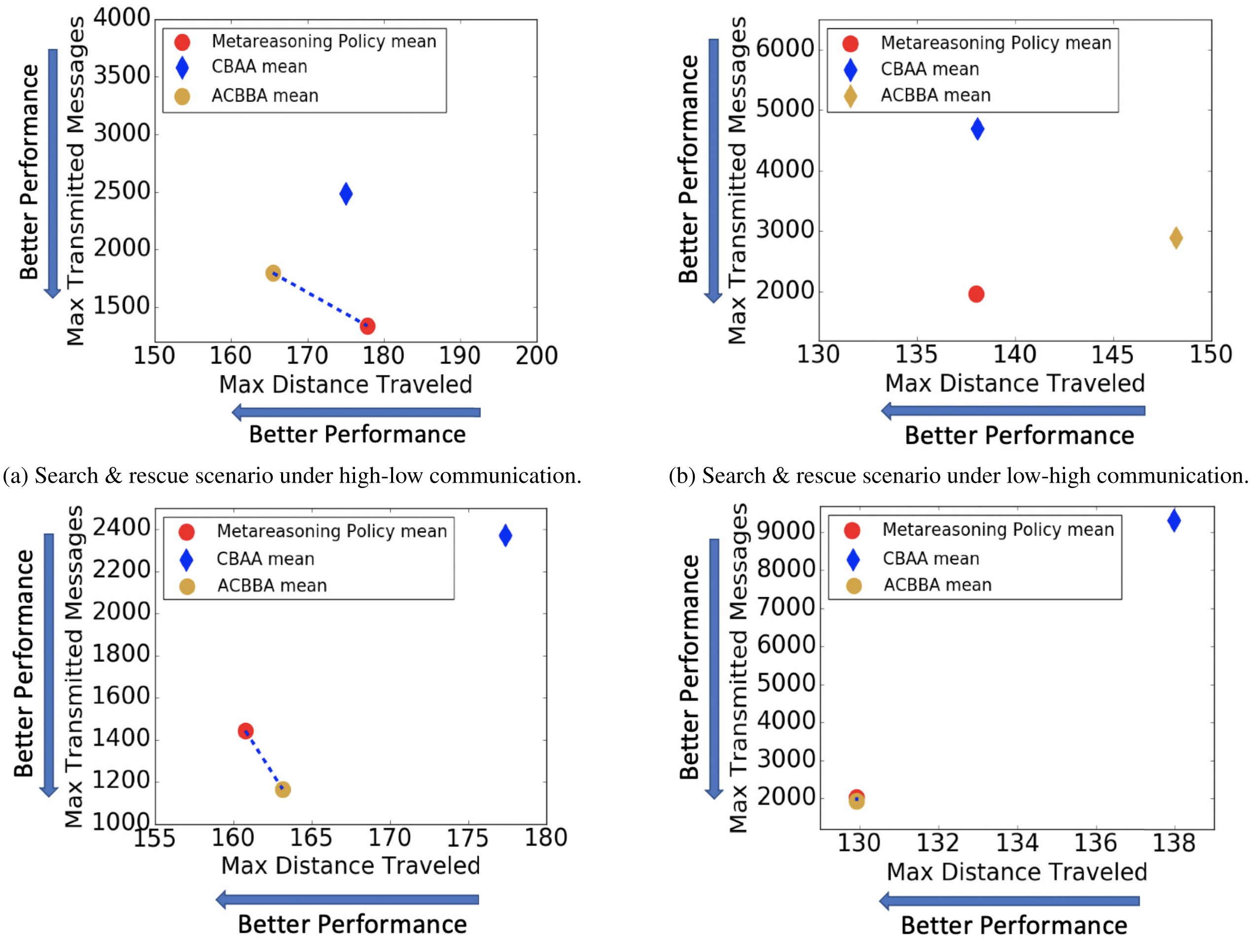
Fire Monitoring Scenario. Ship Protection Scenario.

FIGURE 7. Metrics trade-off analysis results for the fire monitoring (left) and ship protection (right) scenarios. On the left, we show the non-dominated solutions (best performing algorithms on average) with respect to the max number of transmitted messages and/or max distance traveled. On the right, we show the non-dominated solutions with respect to the max number of transmitted messages and/or min distance to the ship. Non-dominated solutions are connected by lines colored differently for each sensitivity threshold value tested. Darker lines signal higher communication. Blue arrows indicate the direction in which system performance improves for each performance metric.

For the ship protection scenario, we note that PI, ACBBA and DHBA were the best performing algorithms on average at high and low communication. At high communication, we note that (PI, ACBBA) shows statistical difference in max distance traveled and number of messages transmitted, while (PI, DHBA) does not exhibit statistical difference in max distance traveled. Amongst these algorithms, we choose PI over ACBBA since it maximizes min distance to ship. Similarly, at low communication, we choose DHBA over ACBBA due to its better performance with respect to min distance to ship.

Since our metareasoning policy involves switching from a multiple task allocation algorithm (i.e. ACBBA) to a single task allocation algorithm (i.e. CBAA) and vice-versa, each agent will continue performing its current task at the time of switching, but its sequence of tasks is reset. A new single task or sequence of tasks is then computed using the new task allocation algorithm. At low communication, multiple agents may pursue the same tasks initially. However, as agents travel in the same direction, they may gain enough communication between them to reach consensus on the remaining tasks. If not, this may lead to some repeated tasks.

In addition, depending on the location of targets, using a single task allocation algorithm may not be as efficient as using a multiple task allocation algorithm, particularly in instances in which an agent is closer to multiple targets while another agent is far away from these targets. With a single task allocation algorithm, the far away agent may be tasked with visiting at least one target and may travel towards its location unnecessarily since the agent closer to this target may win this task ultimately.



(a) Search & rescue scenario under high-low communication.

(b) Search & rescue scenario under low-high communication.

(c) Fire monitoring scenario under high-low communication.

(d) Fire monitoring scenario under low-high communication.

FIGURE 8. Trade-off analysis for the metareasoning policy and each individual algorithm used in the policy. Best expected performing methods are connected by a line and are displayed as shaded circles. Methods with sub-optimal solutions are displayed as shaded diamonds.

VII. RESULTS

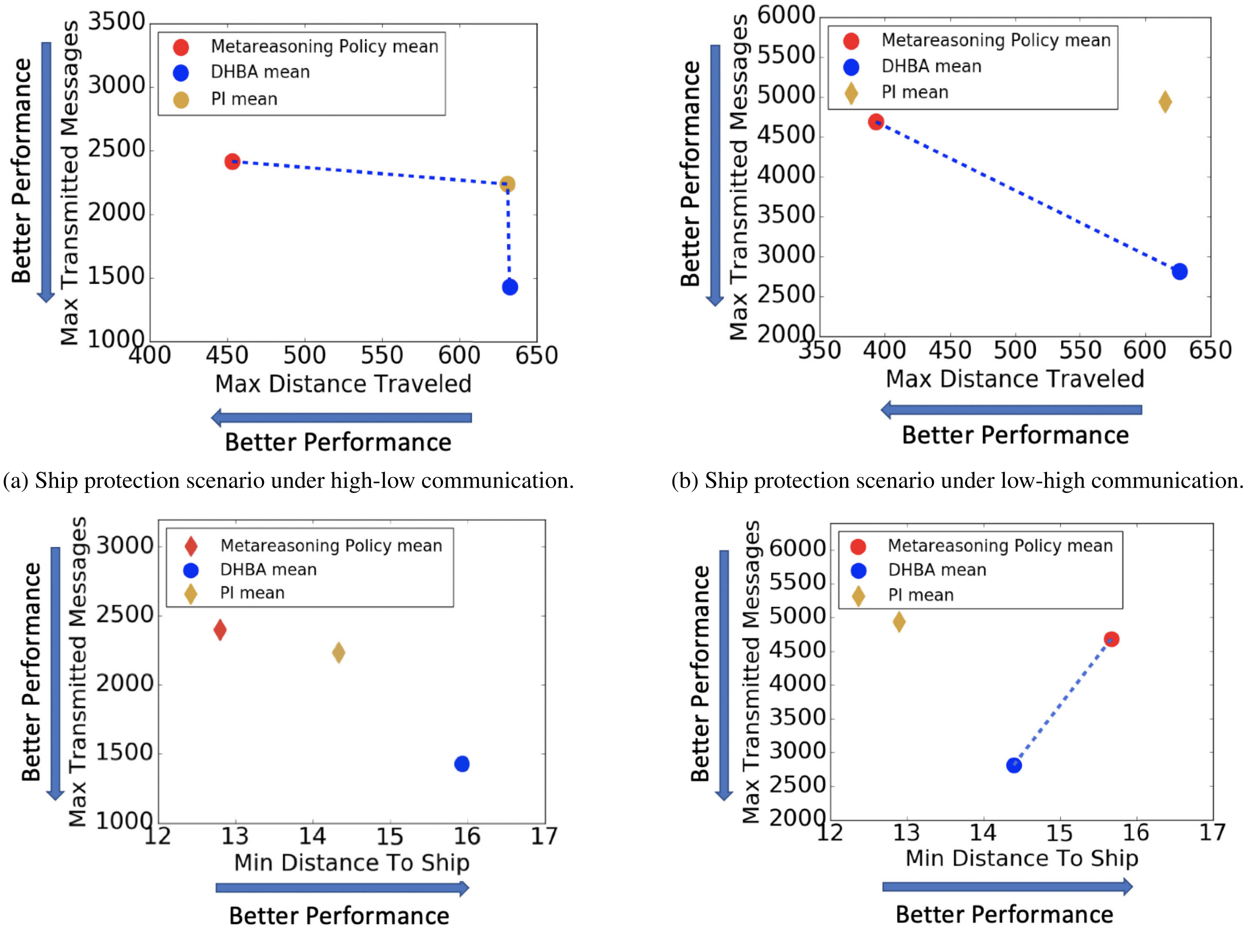
In Figs. 8 and 9, we show our compiled results for the two communication switching conditions tested and each type of scenario. We use trade-off analysis to demonstrate the difference in performance between the metareasoning policy and each individual algorithm used in the policy with respect to the metrics chosen: max distance traveled, max number of transmitted messages, and min distance to the ship (applicable only to the ship protection scenario).

We make a few important observations from these results. First, for the search & rescue scenario, the policy was the best expected performing strategy under low to high communication, but under high to low communication, it only performed better in terms of max number of transmitted messages. Thus, we obtained a trade-off between ACBBA and the metareasoning policy.

In the fire monitoring scenario, the metareasoning policy provided a trade-off with ACBBA under high to low communication. In this communication scenario, the policy obtained the best expected performance in terms of max distance traveled while ACBBA resulted in a slightly lower number of transmitted messages. Under low to high communication, both ACBBA and the policy, are the non-dominated expected solutions. A possible reason for this is the target distribution, in which new fire targets appear near other targets. Agents can get an assignment of multiple nearby targets at once using ACBBA; thus, switching their allocation to a single-task assignment like CBAA may have spread out agents towards targets emerging in different fire clusters, resulting in a larger max distance traveled.

For the ship protection scenario, the policy performed better on average with respect to max distance traveled compared to PI and DHBA. With respect to min distance to ship, the policy only performed better on average under low to high communication. Thus, the policy was more effective in terms of minimizing the max distance to track adversarial targets while DHBA was more effective in terms of maximizing the min distance to ship.

The results obtained demonstrate that on average, the policy outperforms running a single algorithm more



(a) Ship protection scenario under high-low communication.
(b) Ship protection scenario under low-high communication.
(c) Ship protection scenario under high-low communication.
(d) Ship protection scenario under low-high communication.

FIGURE 9. Trade-off analysis for the metareasoning policy and each individual algorithm used in the policy for the ship protection scenario. The analysis with respect to max distance traveled and messages transmitted is shown in the top row. The analysis with respect to min distance to ship and messages transmitted is shown in the bottom row. Best expected performing methods are connected by a line and are displayed as shaded circles. Methods with sub-optimal solutions are displayed as shaded diamonds.

consistently under low to high communication scenarios. Moreover, when agents experienced low communication at the start, metareasoning proved to be more effective when communication improved at a later time. However, when agents started with high communication, changing their task allocation scheme seemed to be less effective. A reason for this may be that agents can reach consensus early on if they are able to communicate successfully at the beginning and may not benefit from changing their already computed solutions as communication degrades. Note that agents perceive low communication as they initiate communication, however this is considered as a transient state of communication and is not explicitly included as part of the overall communication scenario. Also, observe that the max number of transmitted messages for CBAA was consistently larger than the max number of transmitted messages obtained for the other two strategies. This can be explained by scenarios in which agents were not able to receive messages about completed tasks. For this reason, CBAA iterations continued to be executed until at least one agent successfully received information about the completion status of all tasks.

TABLE 4. Average percentages of repeated and missed tasks obtained by the policy and each individual algorithm used in the policy for the search & rescue scenario under high to low communication.

Algorithm	Avg. percentage of repeated tasks (%)	Avg. percentage of missed tasks (%)
CBAA	3.07	0.15
ACBBA	32.5	0.0
Metareasoning Policy	4.98	0.0

In addition to the trade-off plots, results for the average percentage of tasks completed by more than one agent (i.e., repeated tasks) as well as the average percentage of missed tasks are summarized in Tables 4. and 5. for the search & rescue scenario and in Tables 6. and 7. for the fire monitoring scenario. It can be observed that using only CBAA results in a nonzero percentage of missed tasks on average. This was not the case for ACBBA or the policy. Using only ACBBA resulted in a larger percentage of repeated tasks on average compared to using only CBAA or the policy.

TABLE 5. Average percentages of repeated and missed tasks obtained by the policy and each individual algorithm used in the policy for the search & rescue scenario under low to high communication.

Algorithm	Avg. percentage of repeated tasks (%)	Avg. percentage of missed tasks (%)
CBAA	2.59	0.33
ACBBA	44.86	0.0
Metareasoning Policy	3.56	0.0

TABLE 6. Average percentages of repeated and missed tasks obtained by the policy and each individual algorithm used in the policy for the fire monitoring scenario under high to low communication.

Algorithm	Avg. percentage of repeated tasks (%)	Avg. percentage of missed tasks (%)
CBAA	4.70	0.03
ACBBA	33.78	0.0
Metareasoning Policy	8.34	0.0

TABLE 7. Average percentages of repeated and missed tasks obtained by the policy and each individual algorithm used in the policy for the fire monitoring scenario under low to high communication.

Algorithm	Avg. percentage of repeated tasks (%)	Avg. percentage of missed tasks (%)
CBAA	1.66	2.98
ACBBA	38.17	0.0
Metareasoning Policy	4.4	0.0

VIII. RELATED WORK

The following subsections describe related work in which the concepts of metareasoning and decentralized task allocation in MAS have been explored.

A. METAREASONING

Metareasoning research encompasses various approaches to reason about one's own thinking, memory and processing in order to control different aspects of reasoning such as strategy selection and allocation of resources. In regards to controlling strategy selection, metareasoning has been framed as the problem of automated algorithm-switching in computer science [27], [28].

In the context of cooperative MAS, metareasoning approaches have been applied to coordinating the agents' behavior, bringing new challenges as a result of agents performing additional reasoning from which benefits gained might depend on the reasoning and behaviors of other agents [29]. For instance, Raja and Lesser [8] framed multi-agent metareasoning as a decentralized coordination problem in which agents maintain a model of each other's meta-level control and coordinate the use of their reasoning resources. Sleight and Durfee [31] investigated an organizational design approach to coordinate both the agents' behavior and their reasoning by identifying high-performing behavioral patterns and prohibiting agents from reasoning about behaviors counter to these patterns. The effectiveness of metareasoning was shown in [9] for coordinating a team of agents in a tornado tracking application, while [13] applied a centralized controller to coordinate agents with limited communication among them.

B. DECENTRALIZED TASK ALLOCATION

Many of the existing decentralized task allocation approaches are consensus-based auction methods, in which agents place bids on tasks and each agent acts as auctioneer and bidder. A comprehensive survey of these methods can be found in [14]. These approaches include CBAA and CBBA as well as its asynchronous version, ACBBA [15], [16]. Shown to have outperformed CBBA in various problem instances, the PI algorithm [20] does consensus in the same way as CBBA but uses a different valuation function to compute task bids. To improve robustness to dynamic environments and robot failures, Najanath and Gini [10] proposed an auction approach in which each task is treated separately and independently from other tasks.

Another class of decentralized task allocation methods consists of optimization-based approaches, divided into deterministic or stochastic optimization based approaches. Ghassemi and Chowdhury [33] proposed a deterministic optimization based approach in which the task allocation problem is posed as a maximum-weighted matching of a bipartite graph. Another deterministic optimization approach is the DHBA [17], which uses the Hungarian algorithm [34] to solve the task allocation problem and, unlike CBAA, it replaces the auction phase with solving the task assignment problem via the Hungarian method on a cost matrix.

Examples of stochastic optimization-based approaches are the stochastic ant-colony optimization algorithm proposed in [11] and the decentralized GA presented in [36]. The decentralized GA was built as an extension of the GA approach presented in [37] for decentralized systems. Under full communication availability, Patel [36] showed that the decentralized GA outperformed CBBA in a number of problem instances of a rescue scenario. However, it was also shown that the performance of GA degrades significantly as communication quality decreases.

This is not an exhaustive list of task allocation algorithms available in literature. Thus, we refer the reader to [38] and [39] for more comprehensive surveys of task allocation algorithms.

With respect to communication-aware planning and task allocation algorithms in which communication constraints are explicitly considered, a survey of methods that jointly optimize communication and navigation is presented in [3]. For instance, Ponda *et al.* [7] presented a modified CBBA algorithm to predict the network topology and propose relay tasks to repair connectivity violations. In [4], Ponda *et al.* proposed a deconfliction protocol that the mission control center applies to distribute tasks among sub-networks that have low communication between them. Rantanen *et al.* [5]

investigated the performance of ACBBA and explored several network-based configurations to reduce performance degradation caused by varying communication. Although many advances have been made in modifying existing planning algorithms to perform better under limited communication, most multi-robot coordination algorithms are developed under the assumption of perfect communication, which is why in this work, we do not consider communication constraints, and instead, we seek to provide insights into the benefits of switching between existing algorithms with the goal of improving system performance.

In previous work [2] we performed a large set of statistical tests to learn how different multi-agent task allocation algorithms perform across different communication scenarios. We leverage these results to help train our proposed metareasoning approach. In addition to the scenarios considered in [2], our current work includes two additional types of scenarios: the fire monitoring and ship protection scenarios. Similar to our approach, Amorim *et al.* [48] presented an empirical assessment of different swarm-GAP algorithms to solve the task allocation problem in dynamic scenarios. However, we consider varying communication instead of onboard sensor failure or loss of team members. We conduct experiments to assess the performance of the selected algorithms for these additional scenarios and identify the best performing algorithm for each scenario under various levels of communication.

IX. CONCLUSION

In this paper, we describe a metareasoning policy that a team of agents can execute to make effective meta-level control decisions based on the communication availability in the environment. Our policy was tested in various types of scenarios with different types of task allocation problems. We demonstrated that using the policy can lead to gains in performance or trade-offs in terms of max distance traveled and max number of messages transmitted compared to running a single fixed strategy. Within the LTL framework, this policy can be naturally extended to take into account additional environment features and reactive behaviors using LTL specifications.

In addition, with respect to communication, we would like to investigate the effects of more complex communication scenarios, for instance, scenarios in which only certain regions of the environment experience drops in communication quality. With respect to the metareasoning policy proposed, one possible extension is to include in the portfolio of task allocation algorithms considered, those that account for communication constraints and investigate the benefits of switching between such. This can also help us address questions on how to better utilize information exchanged among the agents and how to improve consensus under low communication. With respect to metareasoning, future work can be done to not only execute a prescribed policy, but also modify the policy or generate new policies online. Another important direction for future work would be to analyze the scalability of the task allocation algorithms for a larger number of agents.

APPENDIX A
DETERMINATION OF COMMUNICATION THRESHOLD VALUE

To identify the value of c_t, we ran 15 experiments of the visit scenario described in [2] with 25 targets and 2 agents for a single communication link. Since the communication estimate is independent of the type of scenario, we limited the experiments to this scenario. Initial locations for agents and targets were randomly generated. We varied the sensitivity threshold values from -75 dB to -25 dB in increments of -10 dB. We computed the average over each agent's communication estimates for all time steps. We assume that communication estimates make for a good representation of the communication quality since we expect high quality communication for lower sensitivity threshold values according to the Rayleigh Fading model and hence high communication estimates. Similarly, higher sensitivity threshold values produce low quality communication and hence low communication estimates. We computed the communication estimate by taking the average over the estimates obtained from all the agents. We then computed the mean estimate from all 15 experiments for each sensitivity threshold value. Fig. 10 shows that sensitivity threshold values above -35 dB corresponded to mean communication estimates below 0.2. Thus, c_t was set to 0.2.

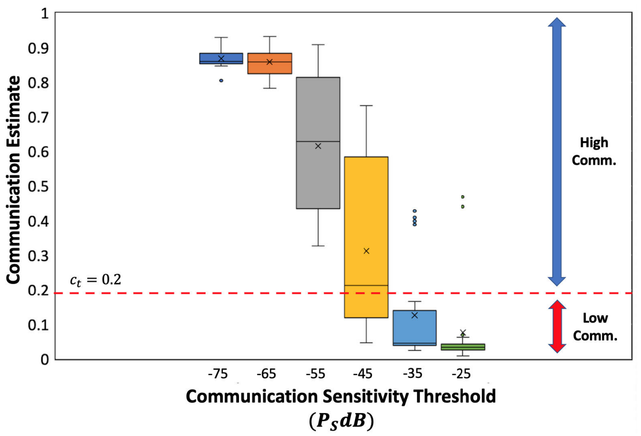


FIGURE 10. Communication estimate values are shown for different sensitivity threshold values. Segmented line indicates the communication estimate threshold value.

To simulate changes in communication, we first determined the ranges of values for γ corresponding to high and low communication, respectively, based on c_t. Generally, values of γ range from 2 (less cluttered environments) to 5 (more obstructed areas) [53]. Thus, we ran the same 15 experiments described above with the difference that we fixed P_S to -65 dB and instead varied γ from 2 to 5 in increments of 0.25. We used these experiments to identify the ranges of values for γ that would likely result in communication estimates above and below c_t. Fig. 11 shows that values from 2.0 to 3.0 resulted in mean communication estimates above c_t while values from 4.5 to 5.0 resulted in mean communication estimates below c_t.

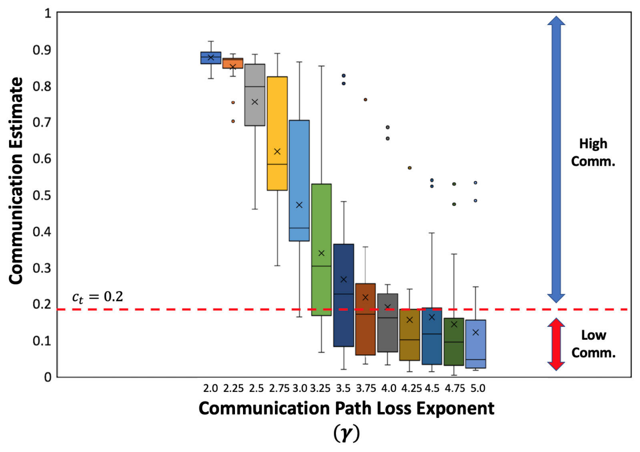


FIGURE 11. Communication estimate values are shown for different path loss exponent values. Segmented line indicates the communication estimate threshold value.

APPENDIX B
DETERMINATION OF WEIGHTING COEFFICIENTS FOR SHIP PROTECTION SCENARIO

Unlike the other scenarios, the ship protection scenario uses a multi-objective function (18) to minimize max distance traveled while driving agents to track targets closer to the ship and continuously search cells in the workspace. The three terms in the objective function are weighted by k_0, k_1 and k_2, respectively. For each algorithm we ran experiments to determine the weights that exhibit the best performance on average with respect to the minimum distance to the ship amongst all targets. We chose this metric since protecting the ship is the primary goal in this scenario.

Assuming perfect communication, we generated 40 samples for (k_0, k_1) using the Latin Hypercube Sampling method [54]. We sampled k_0 and k_1 from [10, 1000]. This range was determined empirically by analyzing the agents' behavior at extreme values of k_0 and k_1 and verifying that their behavior was different. At $k_0 = 10$ and $k_1 = 1000$, agents often visited cells closer to the ship even when they were far away from these cells. At $k_0 = 1000$ and $k_1 = 10$, agents were often assigned to visiting cells that were closer to their own locations. We also observed that $k_2 = 500$ was an effective value to make agents prioritize tracking targets for all values of (k_0, k_1) tested. We performed 50 experiments with 5 agents and 20 targets for each generated sample of (k_0, k_1) and each of algorithm. Initial locations of agents and targets were randomly generated. We set the number of adversarial targets equal to the number of non-adversarial targets.

We computed the average and 95 percent confidence intervals of the chosen metric across all experiments. Amongst the 40 samples, we selected the 3 candidate samples with

TABLE 8. Best performing weights on average for the ship protection objective function with respect to minimum distance to the ship.

	Weights		
Algorithm	k_0	k_1	k_2
CBAA	621	655	500
DHBA	858	258	500
HIPC	109	478	500
ACBBA	483	320	500
PI	215	248	500

TABLE 9. Best algorithm parameters for each algorithm: $\mathcal{I}$ (iteration count) or ($\mathcal{I}$ $\mathcal{B}$) (iteration count, bundle size).

	Algorithm				
Scenario	CBAA	DHBA	HIPC	ACBBA	PI
Fire monitoring	2	2	1	(1, 40)	(1, 2)
Ship protection	4	1	1	(1, 40)	(1, 2)

the highest average minimum distance to the ship. From the selected samples, we kept the one with the smallest confidence interval as this indicated less variance across scenarios. Table 8. shows the best performing weights on average for all algorithms.

APPENDIX C
DETERMINATION OF OPTIMAL ALGORITHM PARAMETERS FOR FIRE MONITORING AND SHIP PROTECTION SCENARIOS

The parameter space of CBAA, DHBA and HIPC contains the max iteration count $\mathcal{I}$, and for ACBBA and PI, it contains both $\mathcal{I}$ and max bundle size $\mathcal{B}$. Values for the $\mathcal{I}$ and $\mathcal{B}$ tuning parameters are chosen as follows: for each $\mathcal{I} \in \{1, 2, 3, 4, 5, 10, 15, 20\}$ and $\mathcal{B} \in \{2, 3, 4, 5, 10, 20, 30, 40\}$. The tuning was done assuming perfect communication. For all the experiments, we used 7 agents, 22 targets for the ship protection scenario and a max of 40 fire targets for the fire monitoring scenario. We set the number of adversarial targets equal to the number of non-adversarial targets in the ship protection scenario.

We show the best tuning parameters for the fire monitoring and ship protection scenarios and each algorithm in Table 9. We set the iteration counts and bundle sizes according to [2] for the search & rescue scenario.

APPENDIX D
DESIGN OF EXPERIMENTS

To compare the performance of the 5 algorithms for the ship protection and fire monitoring scenarios, we used a similar design of experiments and analysis as described in [2]. For each type of scenario, we generated 50 instances with different parameter values shown in Table 10. and Table 11. respectively. We varied the sensitivity threshold P_S in the range $[-25, -75]$ dB in -10 dB increments. Every instance was run at each sensitivity threshold.

For all scenarios, we used a randomized design of experiments. The dimension of the map is $N \times N$ with $N = 100$. We set the agent speed $A_s = 6$ units/s for all scenarios except

TABLE 10. Parameter ranges for instance generation in the search & rescue and ship protection scenarios. The * indicates parameter applies only to the ship protection scenario.

Parameter	Range
Number of agents	[5,10]
Number of target clusters (K)	[1,4]
Ratio of number of targets to agents	[1,4]
Initial locations of agents	([0, 100],[0, 100])
Cluster centers (μ_i)	([0, 100],[0, 100])
Cluster radii (r)	[15,50]
Ratio of number of adversarial targets to non-adversarial targets *	[1,3]

TABLE 11. Parameter ranges for instance generation in fire monitoring scenario.

Parameter	Range
Number of agents	[5,10]
Number of initial fires	[4,10]
Initial locations of fires	([0, 100],[0, 100])
Initial locations of agents	([0, 100],[0, 100])
Wind speed (units/sec)	[5,20]
Wind direction (radians)	[0,2π)

the ship protection scenario. In this scenario, we used a proportional controller and set the max $A_s = 12$ units/s so that agents can track targets before they move out of sensor range. The map is divided into 25 grid cells of size 20 × 20. The detection radius $R_d = 14.14$ units (half length of diagonal of grid cell) while the classification radius $R_c = 8$ units. We set the threshold distance $\delta_T = 0.25$ units.

For the search & rescue and ship protection scenarios, target locations were sampled from a 2D Gaussian Mixture Model described as follows,

$$p_{gmm} = \frac{1}{K}\sum_{i=1}^{K}\mathcal{N}(\mu_i, \Sigma_i), \tag{22}$$

where K is the number of clusters, each with radius r_i, center at μ_i and co-variance $\Sigma_i = \begin{pmatrix} r_i^2 & 0 \\ 0 & r_i^2 \end{pmatrix}$.

For the fire monitoring scenario, initial fire locations were uniformly sampled from [0, 100]×[0, 100]. We set additional parameters including wind speed and wind direction.

For the ship protection scenario, we set the initial location of the ship to (50, 0), heading $\theta_s = 90°$ and speed $v_s = 1.6$ units/sec. In addition, we set the ratio of number of adversarial targets to non-adversarial targets and exclude initial locations of targets that have a distance <40 units from the ship. We set the default speed for all moving targets as 4 units/sec. If adversarial targets are tracked, their speed is set to 12 units/sec.

An instance is defined as one random sampling of the parameters from the ranges mentioned in Table 10. for the search & rescue and ship protection scenarios. For the fire monitoring scenario, we used the parameters from the ranges indicated in Table 11.

For all experiments, we set the Rayleigh fading model parameters as $N = 64$, $d_0 = 1$m, $P_T = 30$ dB, and $P_{L_0} = 40\ dB$ as suggested in [2].

ACKNOWLEDGMENT

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.

REFERENCES

[1] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Auto. Robots*, vol. 44, nos. 3–4, pp. 547–584, Jan. 2019.
[2] S. Nayak, S. Yeotikar, E. Carrillo, E. Rudnick-Cohen, M. K. M. Jaffar, R. Patel, S. Azarm, J. W. Herrmann, H. Xu, and M. Otte, "Experimental comparison of decentralized task allocation algorithms under imperfect communication," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 572–579, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8949727
[3] A. Muralidharan and Y. Mostofi, "Communication-aware robotics: Exploiting motion for communication," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 4, no. 1, pp. 115–139, May 2021, doi: 10.1146/annurev-control-071420-080708.
[4] S. Ponda, J. Redding, H.-L. Choi, J. P. How, M. Vavrina, and J. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *Proc. Amer. Control Conf.*, Baltimore, MD, USA, Jun. 2010, pp. 3998–4003, doi: 10.1109/ACC.2010.5531232.
[5] M. Rantanen, N. Mastronarde, J. Hudack, and K. Dantu, "Decentralized task allocation in lossy networks: A simulation study," in *Proc. 16th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Boston, MA, USA, Jun. 2019, pp. 1–9, doi: 10.1109/SAHCN.2019.8824898.
[6] C. Sabo, D. Kingston, and K. Cohen, "A formulation and heuristic approach to task allocation and routing of UAVs under limited communication," *Unmanned Syst.*, vol. , no. 1, pp. 1–17, Jan. 2014, doi: 10.1142/S2301385014500010.
[7] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H.-L. Choi, and J. P. How, "Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 5, pp. 861–869, Jun. 2012, doi: 10.1109/JSAC.2012.120603.
[8] A. Raja and V. Lesser, "A framework for meta-level control in multi-agent systems," *Auto. Agents Multi-Agent Syst.*, vol. 15, no. 2, pp. 147–196, Aug. 2007, doi: 10.1007/s10458-006-9008-z.
[9] S. Cheng, A. Raja, and V. Lesser, "Multiagent meta-level control for radar coordination," *Web Intell. Agent Systems: Int. J.*, vol. 11, no. 1, pp. 81–105, 2013. [Online]. Available: http://mas.cs.umass.edu/paper/512
[10] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," *Robot. Auto. Syst.*, vol. 58, no. 7, pp. 900–909, Jul. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889010000692
[11] J. Wang, Y. Gu, and X. Li, "Multi-robot task allocation based on ant colony algorithm," *J. Comput.*, vol. 7, no. 9, pp. 2160–2167, Sep. 2012.
[12] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.
[13] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, "Solving transition independent decentralized Markov decision processes," *J. Artif. Intell. Res.*, vol. 22, pp. 423–455, Dec. 2004.
[14] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.
[15] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous decentralized task allocation for dynamic environments," in *Proc. InfotechAerosp.*, Mar. 2011, p. 1441. [Online]. Available: http://hdl.handle.net/1721.1/81434

[16] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Improving the efficiency of a decentralized tasking algorithm for UAV teams with asynchronous communications," in *Proc. AIAA Guid., Navigat., Control Conf.*, Aug. 2010, p. 8421.

[17] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 23–28. [Online]. Available: https://ieeexplore.ieee.org/document/7991447

[18] L. Johnson, H.-L. Choi, and J. P. How, "The hybrid information and plan consensus algorithm with imperfect situational awareness," in *Distributed Autonomous Robotic Systems*. Springer, 2016, pp. 221–233.

[19] L. B. Johnson, H.-L. Choi, and J. P. How, "Hybrid information and plan consensus in distributed task allocation," in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 4888.

[20] W. Zhao, Q. Meng, and P. W. H. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7084641

[21] Y. R. Zheng and C. Xiao, "Simulation models with correct statistical properties for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 51, no. 6, pp. 920–928, Jun. 2003.

[22] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, no. 5, pp. 1977–1997, Sep. 1963, doi: 10.1002/j.1538-7305.1963.tb00955.x.

[23] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, no. 5, pp. 1253–1265, Sep. 1960, doi: 10.1002/j.1538-7305.1960.tb03959.x.

[24] C. Xiao, Y. R. Zheng, and N. C. Beaulieu, "Statistical simulation models for Rayleigh and Rician fading," *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 5, Anchorage, AK, USA, May 2003, pp. 3524–3529, doi: 10.1109/ICC.2003.1204109.

[25] K. E. Baddour and N. C. Beaulieu, "Autoregressive modeling for fading channel simulation," *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1650–1662, Jul. 2005.

[26] C. S. Patel, G. L. Stuber, and T. G. Pratt, "Comparative analysis of statistical models for the simulation of Rayleigh faded cellular channels," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 1017–1026, Jun. 2005, doi: 10.1109/TCOMM.2005.849735.

[27] M. T. Cox, "Metacognition in computation: A selected research review," *Artif. Intell.*, vol. 169, no. 2, pp. 104–141, Dec. 2005, doi: 10.1016/j.artint.2005.10.009.

[28] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *CoRR*, vol. 27, no. 1, pp. 3–45, 2018. [Online]. Available: http://arxiv.org/abs/1811.11597

[29] M. Cox and A. Raja, *Metareasoning: An Introduction*. Cambridge, MA, USA: MIT Press, 2011, pp. 3–14.

[30] S. Karaman and E. Frazzoli, "Vehicle routing with linear temporal logic specifications: Applications to multi-uav mission planning," *Int. J. Robust Nonlinear Control*, vol. 21, pp. 1372–1395, Aug. 2011.

[31] J. Sleight and E. H. Durfee, "Multiagent metareasoning through organizational design," in *Proc. 28th AAAI Conf. Artif. Intell. (AAAI)*, 2014, pp. 1478–1484. [Online]. Available: https://dl.acm.org/doi/10.5555/2892753.2892758

[32] M. A. Finney, "FARSITE: Fire area simulator-model development and evaluation," U.S. Dept. Agricult., Forest Service, Rocky Mountain Res. Station, Fort Collins, CO, USA, 1998, p. 47. [Online]. Available: https://www.fs.fed.us/rm/pubs/rmrs-rp004.pdf

[33] P. Ghassemi and S. Chowdhury, "Decentralized task allocation in multi-robot systems via bipartite graph matching augmented with fuzzy clustering," in *Proc. 44th Design Autom. Conf.*, vol. 2A, Aug. 2018, Art. no. V02AT03A014.

[34] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955.

[35] J. Wang, Y. Gu, and X. Li, "Multi-robot task allocation based on ant colony algorithm," *J. Comput.*, vol. 7, no. 9, pp. 2160–2167, Sep. 2012.

[36] R. Patel, "Multi-vehicle route planning for centralized and decentralized systems," M.S. thesis, Dept. Mech. Eng., Univ. Maryland, College Park, MD, USA, 2019. [Online]. Available: http://hdl.handle.net/1903/25197

[37] H.-J. Choi, Y.-D. Kim, and H.-J. Kim, "Genetic algorithm based decentralized task assignment for multiple unmanned aerial vehicles in dynamic environments," *Int. J. Aeronaut. Space Sci.*, vol. 12, no. 2, pp. 163–174, Jun. 2011.

[38] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004, doi: 10.1177/0278364904045564.

[39] X. Jia and M. Q.-H. Meng, "A survey and analysis of task allocation algorithms in multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2013, pp. 2280–2285.

[40] G. Alexander, A. Raja, E. H. Durfee, and D. J. Musliner, "Design paradigms for meta-control in multiagent systems," in *Proc. AAMAS Workshop Metareasoning Agent-Based Syst.*, 2007, pp. 92–103.

[41] L. Brunet, H.-L. Choi, and J. How, "Consensus-based auction approaches for decentralized task assignment," in *Proc. AIAA Guid., Navigat. Control Conf. Exhib.*, 2008, p. 6839, doi: 10.2514/6.2008-6839.

[42] H. W. Kuhn, *The Hungarian Method for the Assignment Problem*. Berlin, Germany: Springer, 2010, pp. 29–47, doi: 10.1007/978-3-540-68279-02.

[43] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: A software toolbox for receding horizon temporal logic planning," in *Proc. HSCC*, 2011, pp. 313–314, doi: 10.1145/1967701.1967747.

[44] J. I. Smith, "A computer generated multipath fading simulation for mobile radio," *IEEE Trans. Veh. Technol.*, vol. VT-24, no. 3, pp. 39–40, Aug. 1975. [Online]. Available: https://ieeexplore.ieee.org/document/1622250

[45] D. J. Young and N. C. Beaulieu, "The generation of correlated Rayleigh random variates by inverse discrete Fourier transform," *IEEE Trans. Commun.*, vol. 48, no. 7, pp. 1114–1127, Jul. 2000. [Online]. Available: https://ieeexplore.ieee.org/document/855519

[46] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," *J. Comput. Syst. Sci.*, vol. 78, no. 3, pp. 911–938, May 2012, doi: 10.1016/j.jcss.2011.08.007.

[47] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 2002, pp. 1327–1332.

[48] J. C. Amorim, V. Alves, and E. P. de Freitas, "Assessing a swarm-GAP based solution for the task allocation problem in dynamic scenarios," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113437.

[49] S. He, J. Chen, and Y. Sun, "Coverage and connectivity in duty-cycled wireless sensor networks for event monitoring," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 475–482, Mar. 2012.

[50] K. S. V. Narasimha and M. Kumar, "Ant colony optimization technique to solve the min-max single depot vehicle routing problem," in *Proc. Amer. Control Conf.*, Jun. 2011, pp. 3257–3262.

[51] I. R. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot teaming for search and rescue," *IEEE Pervas. Comput.*, vol. 4, no. 1, pp. 72–78, Jan. 2005.

[52] A. Koubaa, *Robot Operating System (ROS): The Complete Reference*, vol. 2, 1st ed. Cham, Switzerland: Springer, 2017. [Online]. Available: https://www.springer.com/gp/book/9783319549262

[53] T. S. Rappaport, *Wireless Communications: Principles and Practice*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[54] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979.

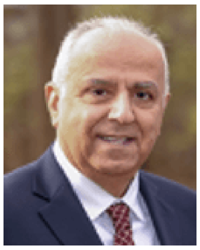
[55] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

ESTEFANY CARRILLO received the B.S. and M.S. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 2012 and 2017, respectively, where she is currently pursuing the Ph.D. degree in aerospace engineering. She is currently a Research Assistant with the Department of Aerospace Engineering, UMD, under the supervision of Dr. Huan Xu. Her research interests include the use of formal methods and hybrid systems theory in the design of verifiable controllers for complex high-level tasks, and control of multi-agent systems.

SUYASH YEOTIKAR received the B.E. degree (Hons.) in computer science from the Birla Institute of Technology and Science at Pilani, Pilani, India, in 2016, and the master's degree in robotics from the University of Maryland (UMD) at College Park, College Park, MD, USA, in 2020. He is currently a Research Assistant with the Department of Aerospace Engineering, UMD, under the supervision of Dr. Huan Xu.

SHARAN NAYAK (Graduate Student Member, IEEE) received the B.E. degree in electronics and communications engineering from Visvesvaraya Technological University, India, in 2009, the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2011, and the M.S. degree in aerospace engineering from the University of Maryland (UMD) at College Park, College Park, MD, USA, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Aerospace Engineering, under the supervision of Dr. Michael Otte. He was a Software Engineer at Garmin International Inc., Olathe, KS, USA, from 2012 to 2016, and a Senior Professional Staff and an Algorithm Developer at the Johns Hopkins Applied Physics Laboratory, Laurel, MD, USA, from 2016 to 2018. His research interest includes motion planning of single and multi-agent autonomous systems. He was a recipient of the Clark Doctoral Fellowship.

MOHAMED KHALID M. JAFFAR received the B.Tech. and M.Tech. degrees in aerospace engineering from the Indian Institute of Technology Madras, Chennai, India, in 2018. He is currently pursuing the Ph.D. degree with the Department of Aerospace Engineering, University of Maryland at College Park, College Park, MD, USA. He is currently a Research Assistant with the Department of Aerospace Engineering, University of Maryland at College Park, under the supervision of Dr. Michael Otte. His research interest includes the use of control theory for motion planning of aerial robots.



SHAPOUR AZARM received the B.S. degree from the University of Tehran, Tehran, Iran, in 1977, the M.S. degree from George Washington University, Washington, DC, USA, in 1979, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 1984, all in mechanical engineering. He joined the University of Maryland at College Park, College Park, in 1984, where he is currently a Professor of mechanical engineering. His research interests include engineering optimization, and decision-making models and methods. He is a fellow of the ASME and a Senior member of AIAA. He is also a Review Editor and a Senior Advisor of (journal of) *Structural and Multidisciplinary Optimization*. He was the Editor-in-Chief of the *Journal of Mechanical Design* (ASME Transactions) and an Associate Editor of (journal of) *Mechanics-Based Design of Structures and Machines*.

JEFFREY W. HERRMANN received the B.S. degree in applied mathematics from the Georgia Institute of Technology, Atlanta, USA, in 1990, and the Ph.D. degree in industrial and systems engineering from the University of Florida, Gainesville, USA, in 1993. He is currently a Professor with the University of Maryland at College Park, College Park, USA, where he holds a joint appointment in the Department of Mechanical Engineering and the Institute for Systems Research. He has authored the textbook *Engineering Decision Making and Risk Management* (Wiley, 2015). His current research interests include collaborative search and tracking and engineering design decision making. He is a member of IISE, ASME, the Design Society, and INFORMS.

MICHAEL OTTE (Member, IEEE) received the B.S. degree in aeronautical engineering and computer science from Clarkson University, Potsdam, USA, in 2005, and the M.S. and Ph.D. degrees in computer science from the University of Colorado Boulder, Boulder, CO, USA, in 2007 and 2011, respectively. From 2011 to 2014, he was a Postdoctoral Associate with the Massachusetts Institute of Technology. From 2014 to 2015, he was a Visiting Scholar with the U.S. Air Force Research Laboratory. From 2016 to 2018, he was a National Research Council RAP Postdoctoral Associate at the U.S. Naval Research Laboratory. He has been with the Department of Aerospace Engineering, University of Maryland, College Park, MD, USA, since 2018. He has authored over 30 articles. His research interests include autonomous robotics, motion planning, and multi-agent systems.

HUAN XU (Member, IEEE) received the B.S. degree in mechanical engineering and material science from Harvard University, Cambridge, MA, USA, in 2007, and the M.S. and Ph.D. degrees in mechanical engineering from the California Institute of Technology, Pasadena, CA, USA, in 2008 and 2013, respectively. She is currently an Assistant Professor with the Department of Aerospace Engineering and the Institute for Systems Research, University of Maryland at College Park, College Park, MD, USA. She is also a member of the Maryland Robotics Center and a Subject Matter Expert of the UMD UAS Test Site. Her research interest includes the use of formal methods and controls in the design and analysis of unmanned autonomous systems. She is a member of AIAA, AUVSI, and INCOSE.