# DOME: Drone-assisted Monitoring of Emergent Events For Wildland Fire Resilience

Fangqi Liu, Janine Baijnath-Rodino, Tung-Chun Chang, Tirtha Banerjee, Nalini Venkatasubramanian
University of California, Irvine
{fangqil2,jbaijnat,tungchuc,tirthab,nalini}@uci.edu

## ABSTRACT

We develop a Drone-assisted Monitoring system, DOME, that gathers real-time data for situational awareness in emergent and evolving events. The driving use case for this work is a prescribed burn event (Rx fire), often used to reduce hazardous fuels in forests. DOME coordinates the use of multiple heterogeneous drone platforms to support the observation of emergent physical phenomena (e.g., fire spread) by leveraging domain expert input and physics-based modeling/simulation methods. We propose an executable rule-based system for drone task generation; here, a high-level mission specification utilizes physics-based models for fire spread prediction and automatically generates monitoring instructions with locations, periods, and frequency for drones. DOME integrates algorithms for task allocation (mapping tasks to drones) and flight path planning while considering trade-offs between sensing coverage and accuracy. In addition, DOME will guide in-flight drones to store and upload data under challenged communication settings (out of transmission range, external signal blocking by trees). We evaluate the performance of DOME in real events (with expert-developed burn plans). We test the applicability of the DOME system using simulated Rx burns at the Blodgett Forest Research Station and evaluate our proposed algorithms by comparing their performance with multiple baseline algorithms. Our experiments show the effectiveness of the composite mechanisms in DOME that outperforms other approaches with higher rewards (capturing data of higher quality) and coverage (reduction of missed tasks).

## 1 INTRODUCTION

This paper examines techniques to monitor dynamic and emergent events for community safety. In general, an emergent event is a planned or unplanned scenario with evolving activities occurring in a space over a timespan - often involving humans, nature, and infrastructure. Examples include large sporting events, military activities, or regions impacted by emergencies such as earthquakes, fires, or floods. In mission-critical scenarios, monitoring ongoing events and understanding their evolution is critical to ensure human safety, mitigate property loss and reduce ecosystem impacts.

Wildland fires are an ongoing threat to those living in rural areas and at the wildland-urban interface, where homes, communities, and wildland vegetation meet or intermingle [38]. From 2012 to 2021, the US had an average of 61,289 wildfires, impacting ∼7.4

million acres annually. In addition, reports from the USDA Forest Services indicate rapid growth within the wildland-urban interface (WUI) in the last decade, with an increase in 33% more land area occupied, 97% of which constitute new homes. Multidisciplinary efforts are made to prevent or reduce the impact of WUI fires, including meteorological reports, drought monitoring, vegetation status monitoring, fire suppression actions, and post-fire recovery strategies [3]. Thus, reliable and timely access to information is required to take action under extreme conditions. However, gaining situational awareness is challenging in rural or remote wildlands and WUI communities with limited infrastructure. Existing technologies for monitoring wildland fires and ambient conditions include remote sensing/satellite imagery and in-situ wireless sensor networks [9]. While helpful, these approaches have practical limitations, such as delays, coarse data resolution and maintenance difficulties due to fire damage [38]. Today, advances in sensing, mobility, and computing capabilities have made low-cost aerial sensing technologies such as unmanned aerial vehicles (UAVs) or drones feasible and suitable in gathering data for Wildland fires [19, 33] By serving as "eyes in the sky," data obtained from carefully coordinated drones equipped with sensors have the potential to enable continuous monitoring of mission-critical events.

In this paper, we utilize a driving use case from the wildfire resilience domain to address issues of effective drone-based monitoring for emergent events. We focus on prescribed (Rx) fires, an important tool used by forest services to manage wildland fires today. Rx fires are controlled burns that experts execute under specific weather conditions to reduce hazardous fuels that can cause future wildfires. In addition, Rx burns are beneficial in restoring healthy ecosystems by removing species that threaten the ecosystem, thus promoting the growth of trees, wildflowers, and other plants [6]. However, inherent risks associated with Rx fires include their potential escape from their planned region, which can turn them into wildfires. For example, the 2012 Lower North Fork Escape fire in Schell Creek Range [28] resulted in multiple civilian casualties, destroyed 27 residences, and caused $11.3 million in property damages. Needless to say, fine-grained, real-time monitoring of Rx fires is critical to its safe execution and consequent adoption at scale in WUI communities. With these concerns, we propose a novel system, called DOME, for supporting drone-assisted Rx fire monitoring, in which multiple diverse drones with sensors (e.g., visual and thermal cameras) effectively collect data above the burn sites to monitor user-specified targets. The DOME system utilizes a physics-based model (for fire prediction), user-specified rules, and perceived fire status to automatically develop detailed drone instructions by generating a series of tasks to specify the monitoring locations, time, and targets. Then, DOME plans the flights of multiple drones by developing waypoint sequences to guide them to fulfill tasks. The flight planning aims to address the trade-off between data quality

and coverage by maximizing task accomplishment and improving the acquisition data resolution. DOME can also handle network interruptions between drones and the ground controller (GC) by allowing drones to transmit data in a store-and-upload manner.

Specific contributions of this paper are: i) the design of DOME, a drone-based monitoring system (§2), ii) design of an automatic and flexible task generation procedure to generate drone monitoring instructions based on given physics-based models and user-defined rules (§3), iii) formulation of the multi-drone flight planning (MFP) problem; and a two-step approach to solve it effectively (§4,§5), iv) implementation of the DOME prototype and thorough evaluation of our proposed algorithms in Rx burn use cases (§6).

## 2 PROBLEM DEFINITION AND APPROACH

We begin by providing the context for Rx fire use case. An Rx fire is coordinated by a *burn boss* who serves as an incident commander and makes decisions during the burn. Burn bosses develop burn plans that pinpoint burn sites, time of burn, and ignition strategy; they assess the potential of an escaped fire and strategies to respond in a timely manner. Situational awareness is assimilated from a number of factors - status of wind, fire spread, spot fires, ember transport, and firefighter mobility to low-level observation parameters, such as vegetation status, fire flame length, and fire intensity. Planning drone usage for fire monitoring in a Rx fire is complex; it requires a systematic approach to guide fire parameter observation, drone placement, and task scheduling.

**Opportunities and challenges in Rx fires.** In the past year, we conducted multiple drone surveys during Rx burns within the Blodgett Forest Research Station as shown in Fig 1. We highlight some observations about executing Rx fires that present unique opportunities and challenges for drone usage. First, Rx fires are human-induced and planned; these planned and controlled nature of these burns enables us to access burn site information, which can be used to predict fire behavior using physics-based models to guide real-time monitoring. Second, multiple features must be monitored during Rx fires, including fire rate of spread, flame length, and location of firefighters and ground personnel. These monitoring targets place different demands on sensors, data quality, and observation frequency. In our Rx burn, we used multiple drones (DJI Matrice 300 RTK and DJI Mini 3) with diverse payloads for capturing a variety of information. For example, RGB images are used to localize fires and humans; the multispectral sensor provides information on vegetation health, and the thermal sensor captures data on fire intensity within the firefront. Furthermore, data quality needs also vary - data used for determining fire intensity require higher-resolution thermal imagery compared to that used for fire detection [26]. Similarly, improving RGB image resolution allows us to improve object detection results with more granular object recognition outcomes [16], In practice, there are coverage/accuracy tradeoffs since the drone's coverage will be reduced with a lower altitude flight but with higher spatial resolutions. This tradeoff between higher spatial resolution and a larger field of view is of concern during data collection. Diverse monitoring requirements point to the need for heterogeneous drones that have varying flying speeds and are equipped with different sensors and networking capabilities. In addition, drones may connect to their ground controller (GC) through wireless techniques such as WiFi, proprietary technologies such as DJI's Lightbridge, or cellular networks [42]. These communication technologies vary in data transmission range from 200 to 3000 meters [38]. An associated challenge in wildland scenarios is the lack of cellular network infrastructures in forest regions; vegetation/trees influence signal attenuation and blocking characteristics even when partially available. Drones typically have limited data transmission range [14] and hence lose communication with the GC during flight. All the above challenges demonstrate the need for a novel multi-drone flight planning approach, which aims to plan the motion of diverse drones to continuously monitor multiple features in an emergent event (Rx fire).

**Existing work and limitations.** We assess related efforts in multi-drone planning and drone-based monitoring, which have been explored in many fields, including operations research (OR) [29] artificial intelligence (AI) [40], and robotics [10]. We discuss their suitability and limitations for the Rx Fire monitoring scenario.

In OR literature, multi-drone flight planning problems have historically been generalized to NP-hard vehicle routing problems (VRPs), where vehicles are coordinated to visit a set of locations [29]. Here, popular methods such as branch-and-cut and dynamic programming have been utilized by CPLEX [11] to obtain optimal solutions; but they are computationally expensive and intractable. This has led to a rise in heuristic-based methods, such as tabu search [4], genetic algorithms[17], evolutionary algorithms [34], and two-phase methods [24], which solve VRPs fast, but with suboptimal solutions. To add to the complexity, domain-specific monitoring requirements and other constraints in RxFires are challenging to model, which may lead to diminished situational awareness.

The multi-drone flight planning problem has been generalized to the cooperative multi-agent planning problem (MAP) in the AI community; this approach involves multiple agents that work together towards a specific goal [40]. The state-of-the-art solutions involve modeling MAP instances using Planning Domain Description Language PDDL or Multi-agent PDDL [2], and applying PDDL solvers such as ADP [12] to obtain actions for agents. However, such solvers cannot adequately address potential anomalies during emergent events. This has led to new heuristic-based methods for new domains, from the Rapidly-exploring Random Tree approach for path planning [15] to horizon optimization and model predictive control for cinematography [32]. Other efforts have leveraged auction-based task allocations [41], Markov decision processes [31], and Monte-Carlo tree search [30] for dispatching emergency responders to disaster zones. All of these solutions focus on specific use cases, such as collision-avoidance path planning and dispatching agents to specific destinations. In contrast to Rx fire scenarios, where repetitive monitoring, disconnected network, and data quality issues dominate, such solutions are not directly applicable.

Related literature in robotics focuses on area coverage as a key objective in drone-based monitoring. This is typically cast as the UAV coverage path planning problem (CPP), which directs multiple UAVs to cover a specific area [8]. Subproblems considered in this regard involve the efficient decomposition of non-uniform areas [37], heuristic algorithms [23] for area partition among multiple UAVs, and waypoint generation and path planning for coverage [5]. Although efficient area coverage is critical in path planning, other requirements such as monitoring frequency, priority, and data quality, must be considered to support monitoring for emergent events.
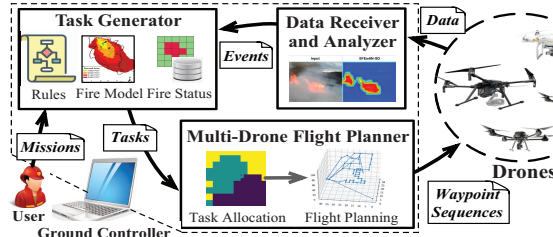
**Figure 1: Our Previous Rx Burns.**



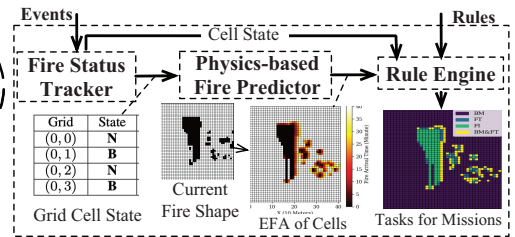**Figure 2: Overview of DOME Framework.**



**Figure 3: Workflow of Task Generator.**

Drone-centric approaches have been explored in many applications showing strong potential in enabling real-time disaster monitoring applications, such as building fire monitoring [27], military missions [7] and environmental monitoring [36]. These settings often require drones to monitor target areas with dynamic threats, and hostile environments [25]. In the context of wildfires, the goals of such monitoring include tracking the fire perimeter [21] and fire intensity deviation [35]. In contrast, automating drone-assisted monitoring in Rx fires requires schemes to monitor multiple features with diverse data quality requirements and deal with the diversity of drone configurations and sporadic network conditions. Furthermore, the availability of the pre-existing burn plan can help guide operation better than a unplanned wildfire scenario.

**Our DOME Approach** This paper proposes DOME, a system to support multi-drone flight planning for monitoring Rx burns, as shown in Fig. 2. DOME considers multiple drones equipped with RGB and/or thermal cameras that continuously fly above the burn site to collect and transmit data to a ground controller (GC) using wireless techniques such as WiFi and DJI's Lightbridge. To launch the DOME system, users (e.g., burn boss) need to provide their monitoring requirements, consisting of a series of *missions*. A *mission* is defined by the following: (i) *mission type*, which describes the type of information we should obtain from sensing data; (ii) *period*, which provides a desired frequency to capture data for the use; and (iii) *significance*, which enables the prioritization of missions. In this work, we consider four types of missions which are: i) **Burn site resource monitoring ($\mathbb{BM}$)**: localizing firefighters and checking the state of equipment at the burn site; ii) **Fire detection ($\mathbb{FD}$)**: detecting fires; iii) **Fire tracking ($\mathbb{FT}$)**: tracking fire spread near the fire front; and iv) **Fire intensity inspection ($\mathbb{FI}$)**: checking the intensity and the flame length of burning fires. This, combined with period and significance details, enables users to specify complex requirements in monitoring, e.g., check fire intensity ($\mathbb{FI}$) every 3 minutes or prioritize fire tracking ($\mathbb{FT}$) to prevent escaped fires.

However, missions only provide coarse monitoring specifications; they do not give detailed information about where (target locations) and when to monitor. To address this issue, DOME leverages a *task generator* component (in Fig. 2) to automatically create a series of *tasks* that are defined by a monitoring area (*location*) and duration (*start/end time*) for an associated mission. Tasks are based on the currently observed fire status, user-defined rules, and a fire prediction model. The generated tasks are given to the *multi-drone flight planner* component (in Fig. 2), which schedules drone flight paths to fulfill the given tasks in two steps: task allocation and flight planning. The task allocation maps all tasks to drones, considering their heterogeneity in mobility, sensing, and networking capabilities. Then, the flight planning step generates a

*waypoint sequence* for each drone, which contains a series of waypoints (3D coordinates) to visit. In this step, DOME additionally considers potential network disconnections, during which drones must decide when to store/upload data, and the tradeoff between data quality and coverage to improve task accomplishment and acquisition data resolution. In this work, we assume that drones will be operating under adverse networking conditions (i.e., disconnections), and thus will store collected data until connection can be established with a ground controller (GC), then transmit data. The GC, comprised of a *data receiver and analyzer*, collects and processes the data, which discloses information required for the diverse missions. This information is then cast as an *event*, which specifies state information, location and time. All events will be reported to a *task generator*, which utilize them to generate tasks for drones based on user-specified rules and physical models.

Considering the intermittent network connection and dynamics of the fire status, DOME periodically plans drone flights for a predetermined duration, called an epoch. To avoid incomplete tasks throughout epochs, we set the epoch length to be a common multiple of the periods for 4 types of missions. At the beginning of each epoch, the task generator generates tasks in this epoch, and our *flight planner* plans the flights of drones to fulfill these tasks. We require that all drones connect with the GC to obtain new flight plans (or return for charging) at the end of each epoch.

## 3 PHYSICS-INSPIRED TASK GENERATOR

In DOME, we introduce a physics-inspired rule-based framework for drone task generation. The entire emergent event duration is divided into epochs, and at the beginning of each epoch, the task generator generates tasks that drones should execute in this epoch. This procedure is called time-driven mode task generation. Each generated task is characterized by a mission, the observation area, and its start and end times. To illustrate the task generation component in DOME, we use the Rx fire driving use case. Here, the task generation procedure creates the spatial-temporal requirements for four missions in §2: Fire detection ($\mathbb{FD}$), Burn site resource monitoring ($\mathbb{BM}$), Fire tracking ($\mathbb{FT}$), and Fire intensity inspection ($\mathbb{FI}$). Fig. 3 illustrates the workflow of the DOME task generator for Rx fires with three components: i) a fire state tracker captures the current state and records the updates of burn site status (based on events reported by the data analyzer), ii) a fire predictor that predicts the evolution of the event using physics-based fire models (FARSITE [18]) and iii) a rule engine with user-specified rules and a rule interpreter (PyKnow [1]) for generating tasks based on the current state (fire status) and future state (prediction results). The rule engine specifies and executes production rules with an 'If' segment indicating an event occurrence with *facts* (e.g., a fire is

detected) and a 'Then' segment with the triggered *actions* (e.g., 'add task') or consequent facts. Two segments are connected by '⇒,' and all facts and actions are expressed by predicates in first-order logic.

**Fire status tracker.** To represent the fire status, we partition the whole burn site **G** into multiple non-overlapping square grids and track the state of each grid cell $g \in \mathbf{G}$. The state $s$ of each $g \in \mathbf{G}$ is one of {**UK, B, NB, BO**}, where Unknown (**UK**) denotes no data has been received for this cell, Burning (**B**) indicates fires are detected within it, Not burning (**NB**) indicates that the fire hasn't arrived in this cell and Burnt out (**BO**) implies that fires within it have burnt out. Our fire status tracker records the state of each $g \in \mathbf{G}$ at each timestamp using $State(g, t)$ in our database. The state of a grid cell $g$'s is updated as our data analyzer reports events related to the existence of fire or no fire at $g$. These events are expressed using facts $Fire(g, t)$ and $NoFire(g, t)$ and state updates are denoted using $ShiftState(g, t, s)$, as shown in Fig. 4. Here, Rules **RST-1** and **RST-2** imply that $g$'s changes from **UK** to **B** or **NB**, depending on whether fires are detected. Rules **RST-3** and **RST-4** express the state shift from **NB** to **B** and **B** to **BO**, resulting from the newly detected fires and the fire burnout. Note that **BO** can only transfer from **B** as fire burnout occurs only at cells that detected fire before.

**Fire prediction.** DOME's task rule engine triggers a fire predictor, which executes fire simulations. The predictor first issues a query to obtain the current fire status from our database; the current fire state serves as input to FARSITE [18], a physics-based fire simulator incorporating multiple models for surface fire, crown fire, spotting, etc. The burn site's landscape, weather conditions (e.g., wind speed, moisture), and the shape of ignition fires are also provided a priori to FARSITE since Rx fires are planned. Information on ignition fires (polygons) is derived by extracting the contour of the grid cells in state **B** into polygons. FARSITE simulates fire growth by producing the fire perimeters (polygons) at all specified time intervals. This is mapped into our grid-based burn site to obtain an estimated fire arrival time (EFA) of each grid cell. If s fire does not arrive at a grid cell within the simulation time, we set its EFA as "INF". In each simulation, we compute the EFA of all cells where fire has not arrived (under state **NB**) for the grid cells where fires have already passed, we set their EFA to the fire simulation time.

**Rule-based task generation.** Next, we illustrate the time-driven task generation rules. Prior to the burn, we assume that the state of every grid cell is **UK** (unknown). To bootstrap DOME, the mission $\mathbb{FD}$ is initiated to scan the whole burn site. Note that DOME can allow for external input to DOME regarding fire status; in this case, the mission FD will execute for cells where the fire status is unknown (i.e., in state **UK**). Once burn site scanning is completed, DOME begins to monitor the dynamic status of the burn site. Specifically, three types of monitoring missions are initiated, which are $\mathbb{FI}$ for checking the fire intensity, $\mathbb{BM}$ for monitoring human/equipment status, and $\mathbb{FT}$ for detecting potential fires in the regions where fires are expected to arrive within the epoch. All tasks are periodically executed during the epoch; each mission has a customized period to indicate its user-defined execution frequency. We next introduce some basic notation for DOME's rule-based task generation framework, i.e., facts, actions, and task generation rules in Fig. 4. State transition rules **RST-1** to **RST-4** are applied over time, which updates the state of grid cells continuously, and Rules

**RT-1** to **RT-5** support time-driven task generation. Here, we define fact $Epoch(t)$ to check if the current time $t$ is at the start of an epoch and maintain a boolean variable $Monitor$, which is false initially and turns true when we can initiate monitoring. The action $Add(m, g, st, et)$ generates a task for mission $m \in \{\mathbb{BM}, \mathbb{FD}, \mathbb{FT}, \mathbb{FI}\}$ at $g$ from time $st$ to $et$. We use action $\forall g(GetEFA(g, t))$ to trigger fire simulation and set $EFA(g)$ as the latest computed EFA of $g \in \mathbf{G}$.

---

**Facts**:
- $State(g, t) = s$: it is true if g's state at time t is $s \in \{\mathbf{UK, B, NB, BO}\}$.
- $Fire(g, t)$: it is true if fires are detected at g at time t.
- $NoFire(g, t)$: it is true if no fire is detected at g at time t.
- $Epoch(t)$: it is true if current time t is at the beginning of an epoch.
- $Monitor$: it is true if we have entered the monitoring phase.

**Actions**:
- $ShiftState(g, t, s)$: change g's state to $s \in \{\mathbf{UK, B, NB, BO}\}$ at time t.
- $\forall g(GetEFA(g, t))$: run FARSITE to obtain EFA of all grid cells based on fire status at time t.
- $Add(m, g, st, et)$: add a task for mission m $\in \{\mathbb{BM}, \mathbb{FD}, \mathbb{FT}, \mathbb{FI}\}$ at g with st and et as its start and end times.

**RST-1** : $\forall g\,((State(g, t) = \mathbf{UK}) \wedge Fire(g, t) \Rightarrow ShiftState(g, t + 1, \mathbf{B}))$
**RST-2** : $\forall g\,(State(g, t) = \mathbf{UK} \wedge NoFire(g, t) \Rightarrow ShiftState(g, t + 1, \mathbf{NB}))$
**RST-3** : $\forall g\,((State(g, t) = \mathbf{NB}) \wedge Fire(g, t) \Rightarrow ShiftState(g, t + 1, \mathbf{B}))$
**RST-4** : $\forall g\,((State(g, t) = \mathbf{B}) \wedge NoFire(g, t) \Rightarrow ShiftState(g, t + 1, \mathbf{BO}))$
**RT-1**: $\forall g(\neg Monitor \wedge (State(g, t) = \mathbf{UK}) \wedge Epoch(t) \Rightarrow Add(\mathbb{FD}, g, t, \sim))$
**RT-2** : $\neg Monitor \wedge \forall g(\neg(State(g, t) = \mathbf{UK})) \wedge Epoch(t) \Rightarrow Monitor$
**RT-3** : $Monitor \wedge Epoch(t) \Rightarrow \forall g(GetEFA(g, t))$
**RT-4** : $\forall g(Monitor \wedge (State(g, t) = \mathbf{B}) \wedge Epoch(t) \Rightarrow Add(\mathbb{FI}, g, t, \sim))$
**RT-5** : $\forall g(Monitor \wedge (State(g, t) = \mathbf{NB}) \wedge Epoch(t) \Rightarrow Add(\mathbb{BM}, g, t, \max\{t, EFA(g) - \delta_{ft}\}) \wedge Add(\mathbb{FT}, g, \max\{t, EFA(g) - \delta_{ft}\}, \sim))$

**Figure 4: Task Generation Rules**

Rule **RT-1** generates tasks for mission $\mathbb{FD}$ to cover all grid cells with state **UK** at the start of an epoch. We use the symbol '~' to indicate the end time of this epoch. Rule RT-2 starts the monitoring phase after an initial fire detection ($\mathbb{FD}$) by setting Monitor to true after all grid cells are not under the state **UK**. Then, rule **RT-3** triggers the fire predictor at the start of the epoch to generate the EFA of all cells, given the current fire status. Rule **RT-4** adds tasks for mission $\mathbb{FI}$ at cells under state **B**, and Rule **RT-5** generates a task for mission $\mathbb{BM}$ at each cell under state **NB** before time EFA(g)-$\delta_{ft}$, after which mission $\mathbb{FT}$ starts execution. The threshold $\delta_{ft}$ forces tasks for $\mathbb{FT}$ to execute earlier than the cell's EFA, which addresses the uncertainty in prediction and flying time for drones. To order the execution of rules that may be triggered simultaneously, we prioritize state transition rules as follows (high to low): i) **RST-1** to **RST-4**, ii) rule **RT-3**, and iii) rules **RT-1**, **RT-2**, **RT-4**, and **RT-5**. This forces our task generator to update cell states first, then compute their EFA, and finally generate tasks.

In our framework, we also propose an 'event-drive task update' procedure (using rules **RE-1** to **RE-5** in Fig.9 in Appendix) to support the dynamic update of ongoing tasks during the epoch; meanwhile, drones can adjust their flights to accommodate these updates. This flight adjustment requires stable communication between GC and the drones for task reallocation during the epoch.

## 4 MULTI-DRONE FLIGHT PLANNING

Using the time-driven task generation, we next explore the flight planning procedure. At the start of an epoch, the flight planner

coordinates multiple drones to fulfill all generated tasks within the epoch. In this section, we formulate the multi-drone flight planning problem (MFP) as a combinatorial optimization problem.

## 4.1 Symbols and Notations

In DOME, drones are assigned missions/tasks detailing spatial-temporal sensing monitoring requirements. To this end, waypoint candidates describing specific drone flight paths are proposed.

**Missions and tasks.** We first formulate the missions and generated tasks specified in the previous sections by letting $j \in [1, M]$ denote a mission, which has a corresponding period $p(j)$ and significance $\sigma(j)$. For guiding drones to capture valuable data of high resolution, we evaluate and score data quality for diverse missions by comparing its resolution, measured by Pixel Per Meter (PPM), against certain threshold values proposed in empirical standards [16, 26] (more details are provided in Appendix C). In particular, for each mission $j$, the quality of the data captured by a sensor $s$ is evaluated using a score $SC_s^j(pm)$, which is a function of its PPM value $pm$. Intuitively, data of higher quality (i.e., resolution) will receive higher scores. Table 1 shows a data quality scoring rubric. For example, the cell located at the intersection of the Thermal and $\mathbb{BM}$ represents that if a thermal camera captures data for $\mathbb{BM}$ with PPM value $pm$, then the data's quality score is 0 if $pm < 8.48$, 0.6 if $8.48 \le pm < 10.6$, 0.75 if $10.6 \le pm < 15.2$ and 0.9 otherwise. As RGB images cannot reveal fire intensity (temperature) information, we set 'N/A' in the cell for RGB and $\mathbb{FI}$.

**Table 1: Data quality score under diverse PPMs**

| Mission / Sensor | BM | | FD/FT | | FI | |
|---|---|---|---|---|---|---|
| | PPM | Score | PPM | Score | PPM | Score |
| Thermal | 8.48 | 0.6 | 6 | 0.6 | 12 | 0.6 |
| | 10.6 | 0.75 | 7.5 | 0.8 | 15 | 0.8 |
| | 15.2 | 0.9 | 10.74 | 1 | 21.4 | 1 |
| RGB | 25 | 0.6 | | | | |
| | 62 | 0.85 | 262 | 1 | N/A | N/A |
| | 125 | 1 | | | | |

To realize these missions, a series of tasks $T_i \in \mathbf{T}$ are generated. Each task $T_i$ is defined using the 4-tuple $(m_i, g_i, \phi_i, e_i)$, which specifies a covering area (grid cell) $g_i \in \mathbf{G}$ that a drone must observe during time duration $(\phi_i, e_i)$ for its associated mission $m_i \in [1, M]$. To track the partial completion of periodic tasks, we further split each task $T_i$ into multiple subtasks based on its period $p(m_i)$ and use $T_i^k$ to indicate its $k$th subtasks, with $k > 0$. Then, we define a subtask's release time $r(T_i^k) = \phi_i + (k-1)p(m_i)$, and deadline $d(T_i^k) = r(T_i^k) + p(m_i)$, which are used to represent when the subtask is released or expired, respectively.

**Drones characteristics.** Suppose that drone $d \in [1, D]$ is defined by its maximum data transmission range $rng_d$, flying speed $spd_d$, and is equipped with a set of visibility sensors $sen_d$. In this paper, we assume the communication between drones and the GC is stable as long as the ground controller is within the transmission range $rng_d$ of the drone. We also assume that all sensors in $sen_d$ are fixed on the drone to point downwards and that quantities such as sensor coverage range (CR) and captured image resolution (PPM) can be computed for some height, given sensor configurations. We use Eqs. (1a) and (1b) derived in [13] to find PPM and CR values captured by drone $d$'s sensor $s \in sen_d$ at height $h$, where $FH_s$ and $FV_s$ (degree) indicate its horizontal and vertical field of view (FOV), and $PH_s$ and $PV_s$ (pixels) represent its horizontal and vertical image

resolution. In general, these equations reveal that drones at lower heights can capture more detailed images (higher PPM) but at the cost of a smaller coverage range.

$$PPM_s(h) = PH_s / (2h \times \tan(FH_s/2)) \tag{1a}$$

$$CR_s(h) = 2h \times \min\{\tan(FH_s/2), \tan(FV_s/2)\} \tag{1b}$$

**Waypoint candidate generation.** Given the Rx fire site $\mathbf{G}$, we find potential locations above the burn site for each drone $d \in [1, D]$, to ensure the required coverage of the burn site and capture data for all executable missions. These locations, which we refer to as drone $d$'s *waypoint candidates* (WPCs), are a set of 3D coordinates $\mathbf{W}_d = \{w_i : w_i = (x(w_i), y(w_i), z(w_i))\}$, which are derived based on missions' PPM requirements and drone $d$'s sensing capability. Here we also bound drones' flight height within $[H_{min}, H_{max}]$ to keep their distance from trees/fires and fly legally. To generate $\mathbf{W}_d$, we construct a set of WPCs for each drone $d$'s sensor $s \in sen_d$, and each mission $j \in [1, M]$. For a mission $j$ and a sensor $s$, we let $\mathbf{TH}(j, s)$ denote the set of PPM threshold values for executing mission $j$ using sensor $s$, as shown in Table 1. Then, we deduce the set of potential heights at which sensor $s$ should capture data as $\mathbf{HSet}(j, s) = \{\min(h, H_{max}) : PPM_s(h) \in \mathbf{TH}(j, s), h \ge H_{min}\}$. Next, we generate WPCs to let sensor $s$ cover the whole burn site at each height $h \in \mathbf{HSet}(j, s)$. To do this, we first round its coverage range to the nearest multiple of a burn site's grid size, i.e., $\lfloor CR_s(h)/size(g) \rfloor \times size(g)$ (to ensure fully capturing burn site's grid cells). Then we divide the burn site into squares equal to this rounded coverage range and get a set of WPCs positioned at the center of each of these squares. We repeat this procedure for each sensor $s \in sen_d$ and $j \in [1, M]$ to produce $\mathbf{W}_d$. Note that if $\mathbf{HSet}(j, s)$ is empty for all $s \in sen_d$, drone $d$ cannot execute mission $j$.

In addition, we define $\Lambda_d(w_i)$ with $w_i \in \mathbf{W}_d$, to represent the *network connectivity status* between drone $d$ and the GC at $w_i$. We set $\Lambda_d(w_i) = 1$ if drone $d$ at $w_i$ is connected with the GC, i.e., $rng_d \ge Dis(w_i, g')$ with $g'$ is the location of the GC, and $\Lambda_d(w_i) = 0$ otherwise. We compute drone $d$'s flying time between two waypoints by $FLT_d(w_i, w_j) = (Dis(w_i, w_j)/spd_d) + T_{loi}$, where $Dis(w_i, w_j)$ is the distance between them, and $T_{loi}$ is the loiter time the drone spend at a waypoint to capture and upload data as needed.

## 4.2 Spatial-temporal Factors for Task Execution

The data captured by drones for tasks intrinsically have spatial and temporal components. In general, let $\mathbf{Q} = \{\mathbf{Q}^1, \dots \mathbf{Q}^D\}$ denote waypoint sequences on which the $D$ drones fly to fulfill a set of tasks $\mathbf{T}$ during time duration $[t_0, t^*]$. Each waypoint sequence $\mathbf{Q}^d$ for a drone $d$ is denoted by $\langle w_{q_{(0)}^d}, w_{q_{(1)}^d}, \dots \rangle$ with $w_{q_{(n)}^d} \in \mathbf{W}_d$.

**Spatial factors.** To assess how the data captured by drones can contribute to tasks, we consider its captured spatial location. We define a *data value function* $V_d(w_j, T_i)$ in Eq. (2) which evaluates the value of data that drone $d$ captured at $w_j \in \mathbf{Q}^d$ for task $T_i \in \mathbf{T}$.

$$V_d(w_j, T_i) = \sigma(m_i) \times \max_{s \in sen_d} \left( SC_s^{m_i}(PPM_s(z(w_j))) \times Cov_s(w_j, T_i) \right), \tag{2}$$

where $\sigma(m_i)$ is the $m_i$'s significance value. $PPM_s(z(w_j))$ computes the PPM value of data drone $d$ captured at $w_j$. $SC_s^{m_i}(PPM_s(z(w_j)))$ gives its data quality score for executing mission $m_i$. We also set the coverage correlation $Cov_s(w_j, T_i)$ to 1 if task $T_i$ (at $g_i$) is within the coverage range of drone $d$'s sensor $s$ at $w_j$, and 0 otherwise.

**Temporal factors.** Data captured at a WPC for a subtask is only useful if its collection and upload time are within the subtask's release time and deadline. Thus, we define $\Theta(\mathbf{Q}^d, T_i^k)$ to represent indices in $\mathbf{Q}^d$ indicating the waypoints at which drone $d$ collects valuable data for subtask $T_i^k$ by

$$\Theta(\mathbf{Q}^d, T_i^k) = \{n : n \in [1, |\mathbf{Q}^d|], V_d(w_{q_{(n)}^d}, T_i) > 0,$$
$$AT_d(\mathbf{Q}^d, n), UT_d(\mathbf{Q}^d, n) \in (r(T_i^k), d(T_i^k)]\}$$

where $AT_d(\mathbf{Q}^d, n)$ and $UT_d(\mathbf{Q}^d, n)$ denotes the drone's arrival time and upload time at the $n$th waypoint $w_{q_{(n)}^d}$, in $\mathbf{Q}^d$, respectively. We compute these by $AT_d(\mathbf{Q}^d, n) = \sum_{i \in [0, n-1]} FLT_d(w_{q_{(i)}^d}, w_{q_{(i+1)}^d})$ and $UT_d(\mathbf{Q}^d, n) = \min\{AT_d(\mathbf{Q}^d, i) : i \in [n, |\mathbf{Q}^d|], \Lambda_d(w_{q_{(i)}^d}) = 1\}$. Note that $UT_d(\mathbf{Q}^d, n)$ is the drone $d$'s arrival time at the first waypoint, whose index is no less than $n$, after the drone connects to the GC.

## 4.3 Formulating MFP

We cast our multi-drone flight planning problem (MFP) as a combinatorial optimization problem for generating waypoint sequences $\mathbf{Q}$ for $D$ heterogeneous drones to cooperatively fulfill all tasks $\mathbf{T}$ during $[t_0, t^*]$. Each drone starts at a waypoint $w_0^d$ and needs to return back to depot $w_{dpt}$ in the end. We start by defining the reward for each subtask $T_i^k$ to evaluate the performance of multiple drones executing each subtask given their waypoint sequence $\mathbf{Q}$ by

$$\mathbb{R}(\mathbf{Q}, T_i^k) = \begin{cases} -\beta, & \text{if } \sum_{d \in [1,D]} |\Theta(\mathbf{Q}^d, T_i^k)| = 0, \\ \max_{d \in [1,D]} \left( \max_{n \in \Theta(\mathbf{Q}^d, T_i^k)} V_d(w_{q_{(n)}^d}, T_i) \right), & \text{else.} \end{cases} \tag{3}$$

The reward of a subtask is negative if it is missed, i.e., $\sum_{d \in [1,D]} (|\Theta(\mathbf{Q}^d, T_i^k)| = 0$. In this case, we get a reward $-\beta < 0$ which is a user-specified penalty value for missing a subtask. *In this paper, we let the value of $\beta$ be larger than the maximum mission significance value* to prioritize task accomplishment over data value improvement. Note that the subtask's reward is set to the maximum data value obtained across all data collected by $D$ drones; this is the best value we can obtain from analyzing data under the given time constraints. We then formulate our MFP problem as follows:

$$\max_{\mathbf{Q}} \sum_{T_i^k : t_0 < d(T_i^k) \leq t^*; T_i \in \mathbf{T}} \mathbb{R}(\mathbf{Q}, T_i^k) \tag{4a}$$

$$\text{s.t.} \quad AT(\mathbf{Q}^d, |\mathbf{Q}^d|) \leq t^*, \ \forall d \in [1, D], \tag{4b}$$

$$w_{q_{(0)}^d} = w_0^d, \quad w_{q_{(|\mathbf{Q}^d|)}^d} = w_{dpt}, \ \forall d \in [1, D]. \tag{4c}$$

Our objective function (Eq. (4a)) maximizes the sum of the rewards for all subtasks. We use $\{T_i^k : k > 0, T_i \in \mathbf{T}, t_0 < d(T_i^k) \leq t^*\}$ to represent the set of $T_i$'s subtasks during epoch $[t_0, t^*]$, which contains all subtasks with deadlines between time $t_0$ and $t^*$. Constraint (4b) requires that each drone's flying time does not exceed the end time of the epoch, while constraint (4c) ensures the validity of the waypoint sequence. Note that if we require drones to connect with the GC instead of returning to the depot at the end of the epoch, Eq. (4c) can be replaced with $\Lambda_d(w_{q_{(|\mathbf{Q}^d|)}^d}) = 1$. In general, MFP is an NP-hard problem, which can be proven by reducing the Orienteering problem [22] to a special case of our problem, where there is only one mission with a single PPM requirement and a single drone.

## 5 PROPOSED ALGORITHMS FOR MFP

We solve the MFP problem in two steps and design heuristics as follows: i) task allocation (§5.1), which assigns every generated task to a specific drone; and ii) flight planning (§5.2), where each drone computes its waypoint sequence to fulfill its given tasks.

## 5.1 Step 1: Allocating Tasks to Drones

Our task allocation problem aims to allocate all generated tasks, represented by a set $\mathbf{T}$, to $D$ drones that have diverse mobility, sensing, and networking capabilities. Here each task $T_i \in \mathbf{T}$ is represented by a 4-tuple $(m_i, g_i, \phi_i, e_i)$, which specifies a grid cell $g_i \in \mathbf{G}$ that a drone must observe during time duration $(\phi_i, e_i)$ for its associated mission $m_i$. To optimize the task allocation, an exhaustive search of $D^{|\mathbf{T}|}$ possible task allocations to estimate the drones' obtained reward (using Eq. (4a)) is infeasible given time constraints. Thus, we propose a heuristic to reduce the time taken by all drones to complete all tasks, assuming drones operate concurrently. Assume a set of tasks $\mathbf{T}'$ are assigned to drone $d$, we define a *time utilization* function $\mathbb{U}_d(\mathbf{T}')$ to measure the its time usage for fulfilling $\mathbf{T}'$ by

$$\mathbb{U}_d(\mathbf{T}') = \sum_{j=1}^{M} ET_d(\{T_i : T_i \in \mathbf{T}', m_i = j\})/p(j) \tag{5}$$

where $\{T_i : T_i \in \mathbf{T}', m_i = j\}$ denotes the subset of tasks in $\mathbf{T}'$ for a mission $j$ with period $p(j)$, and $ET_d(\{T_i : T_i \in \mathbf{T}', m_i = j\})$ indicates the task execution time for drone $d$ to complete these tasks once. Additionally, we group all tasks based on their missions since different missions may have different periods. The $\mathbb{U}_d(\mathbf{T}')$ is the sum of the ratios between the execution time of multiple groups of tasks and their periods. It is easy to see that, the lower $\mathbb{U}_d(\mathbf{T}')$ ($<1$) yields a higher probability that drones finish their assigned tasks. Thus, the objective of our task allocation approach is to split all tasks $\mathbf{T}$ into $D$ disjoint subsets $\{\mathbf{T}_1, \dots \mathbf{T}_D\}$ with $\bigcup_{i \in [1,D]} \mathbf{T}_i = \mathbf{T}$, and assign a set of tasks $\mathbf{T}_d$ to drone $d$, such that the maximal time utilization across all drones, i.e., $\max_{d \in [1,D]} \mathbb{U}_d(\mathbf{T}_d)$ is minimized. We propose a utilization-based task allocation (UTA) approach (Alg. 2 in Appendix A) to estimate each drone's time utilization given its assigned tasks and allocate tasks considering the heterogeneous sensing and networking capabilities of drones.

**Spatial task clustering.** Before assigning tasks to drones, we propose to cluster tasks based on their spatial (location) similarity. With this effort, we can speed up the task allocation procedure by assigning a group of tasks (a task cluster) to a drone at each iteration (described later). In particular, we divide the entire burn site into multiple squares of size $\Gamma$ (a multiple of burn site grid size) and generate multiple task clusters $\{\mathbf{C}_1, \mathbf{C}_2, \dots\}$, disjointed sets of tasks with $\mathbf{C}_i \subseteq \mathbf{T}$ and $\bigcup_i \mathbf{C}_i = \mathbf{T}$, where each task cluster contains tasks within the same square and for the same mission.

**Iterative allocation of task clusters.** We heuristically assign a initial task cluster to each drone based on its sensing and networking capabilities. More specifically, to each drone, we assign a task cluster with its executable mission; with the assignment closer to the GC if this drone has shorted data transmission range. We also heuristically select clusters far from those already assigned, aiming to distribute drones over the burn site sparsely. After this, we start iteratively allocate task clusters to drones. In each iteration, all drones estimate their time utilization after receiving each unallocated cluster; subsequently, we assign the cluster to the drone whose time utilization after obtaining the cluster is the lowest.

Next, we illustrate how a drone $d$ computes its time utilization given its assigned tasks $\mathbf{T}'$ in each iteration. We find $ET_d(\{T_i : T_i \in \mathbf{T}', m_i = j\})$ by computing the total time that the drone: (i) flies from its start point to the first waypoint; (ii) traverses a set of waypoints for executing tasks ($\{T_i : T_i \in \mathbf{T}', m_i = j\}$), and (iii) returns for uploading data. Here we only allow drone $d$ to fly at the same height $h$, where it can fulfill mission $j$ and get the maximum coverage using one of its sensors. Then we select the set of waypoints at height $h$ from its WPCs $\mathbf{W}_d$ to cover all these tasks with mission $j$. We estimate the task execution time by considering that the drone follows the Nearest Neighbor approach [20], a 2-approximation algorithm for solving traveling salesman problem, which rules that drones will always fly to the next closest waypoint. In this way, we get $ET_d(\{T_i : T_i \in \mathbf{T}', m_i = j\})$ for each $j \in [1, M]$, and then compute $\mathbb{U}_d(\mathbf{T}')$ following its definition above. The complexity of the time utilization estimation procedure is $O(|\mathbf{W}_c|^2)$ where $|\mathbf{W}_c|$ is the maximum size of the group of generated WPCs at the same height for covering a mission. The complexity of the UTA algorithm is $O(|\mathbf{T}|^2 D |\mathbf{W}_c|^2)$. Alg. 2 in Appendix shows the pseudo-code.

## 5.2 Step 2: Single Drone Flight Planning

After task allocation, each drone plans its flight to fulfill its assigned tasks. In particular, tasks $\mathbf{T}_d$ assigned to drone $d$ at time $t_0$ at $w_0^d$ are used in our single drone flight planning problem to generate a waypoint sequence using WPCs in $\mathbf{W}_d$ in epoch $[t_0, t^*]$ to maximize the total reward. As a special case of our MFP problem with a single drone, this problem is NP-hard. Here, each a series of waypoints are selected based on the drone's status and task completion condition. A Markov decision process (MDP) is used to model this discrete decision-making process while tracking the above information.

**States.** A *state* is defined by a tuple $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle$, where $w \in \mathbf{W}_d$ indicates the drone's current waypoint at time $t \in [t_0, t^*]$. We use $\mathbf{S}(\mathbf{T}_d) = \{(T_i, DL_i, UR_i, SR_i) : T_i \in \mathbf{T}_d\}$ to denote the state of all released tasks for drone $d$. Each task $T_i$ has a deadline $DL_i$ for its ongoing subtask. We use $UR_i$ and $SR_i$ to indicate the maximum data value (defined in Eq. (2)), for $T_i$'s ongoing subtask, of uploaded and stored data respectively. We set the initial state to $\langle w_0^d, t_0, \mathbf{S}_0(\mathbf{T}_d) \rangle$ with $\mathbf{S}_0(\mathbf{T}_d) = \{(T_i, \max(t_0, \phi_i) + p(m_i), 0, 0) : T_i \in \mathbf{T}_d\}$.

**Action.** An action is used to indicate that the drone flies to a waypoint $w' \in \mathbf{W}_d$.

**Transitions.** We use transition function $\mathbb{TX}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w') = \langle w', t', \mathbf{S}'(\mathbf{T}_d) \rangle$ to represent the state update after the drone flies to $w'$, where $t'$ is updated by flying time $FLT_d(w, w')$. For each $T_i \in \mathbf{T}_d$, with $\phi_i \le t'$, we check: (1) If $t'$ passes $T_i$'s subtask's deadline, i.e., $t' > DL_i$, we reset its $UR_i' = SR_i' = 0$ and update $DL_i'$ to the deadline of the newly released subtask. (2) if drone $d$ at $w'$ connects with the GC, i.e., $\Lambda_d(w') = 1$, we update $UR_i' = \max(UR_i, V_d(w', T_i), SR_i)$ and $SR_i = 0$ to model the drone uploading stored and captured data to the GC. Otherwise, we set $SR_i' = \max\{V_d(w', T_i), SR_i\}$ to mimic the drone capturing and storing data.

**Action reward.** We define *action reward* $\mathbb{AR}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$ as the overall change of rewards for subtasks (specified in Eq. (3)) caused by missing subtasks and uploaded data following an action. Given its next state $\langle w', t', \mathbf{S}'(\mathbf{T}_d) \rangle$, we compute the action reward by (1) penalizing any expired subtasks during $(t, t')$ that did not get valuable data by $K = |\{T_i : T_i \in \mathbf{T}_d, t < DL_i < t', UR_i =$

---

**Algorithm 1** DOME Flight Planning (DFP)

**Input:** Initial state ($w_0^d, t_0, \mathbf{S}_0(\mathbf{T}_d)$), ending time $t^*$ and WPCs $\mathbf{W}_d$
**Output:** Drone $d$'s Waypoint sequence $\mathbf{Q}^d$

1  Queue: Group subtasks by deadlines, sort groups from earliest to latest.
2  **while** $t \le t^*$ **do**
3      **if** *STUpdate* **then** Update and reorder the Queue ;
4      **for** *Each group in the Queue* **do**
5          Get all unfinished subtasks in the group
6          **while** *There are unfinished subtasks* **do**
7              Select a **Valid** waypoint $w'$ to cover unfinished subtasks.
8              **if** *No WPC to select* **then break** (Go to next group);
9              **else** Add $w'$ to $\mathbf{Q}^d$; **Transit State**; If *STUpdate* Go to line 3;
10     **if** *Drone's storage is empty* **then**
11         **while** *No Subtask Update* **do**
12             Select a **Valid** waypoint $w'$ to improve reward greedily.
13             **if** *No WPC to select* **then break**;
14             **else** Add $w'$ to $\mathbf{Q}^d$; **Transit State**; If *STUpdate* Go to line 3;
15     **if** $\mathcal{UP}(w)$ *is Valid* **then** Add $\mathcal{UP}(w)$ to $\mathbf{Q}^d$ **else** Add Depot to $\mathbf{Q}^d$;
16     **Transit State**; If *STUpdate*, Go to line 3

---

$0\}|$, by a factor of $-\beta$ and (2) updating increases in data value $\sum_{T_i \in \mathbf{T}_d} (UR_i' - UR_i)$, if the drone connects with the GC at $w'$.

Each episode in our MDP starts with the initial state and ends when it reaches a state with $t \ge t^*$. Our MDP's worst case state-space complexity is $|\mathbf{W}_d|^{(t^* - t_0)/T_{loi}}$ where $|\mathbf{W}_d|$ indicates the number of drone $d$'s waypoint candidates, and $\frac{t^* - t_0}{T_{loi}}$ is the maximal waypoint selection time in an epoch. Due to its exponential state space, we design the DOME flight planning heuristic algorithm (DFP), i.e., the MDP policy to select waypoints at each state. DFP's novel aspects include its validity-checking procedure to ensure timely data upload in a disconnected network and the heuristics for waypoint selection, considering improving task accomplishment and data quality. Alg. 1 shows the pseudo-code of DFP approach.

**Validity checking.** To enable the drone to quickly upload data from the waypoints that are out of the data transmission range, we specify an *uploading point* $\mathcal{UP}(w)$ for each $w \in \mathbf{W}_d$. If drone $d$ at $w$ disconnected with the GC, we set $\mathcal{UP}(w)$ to its closest WPC where the drone is connected; otherwise, we set $\mathcal{UP}(w)$ to itself. Moreover, assuming a drone is at $w$ at time $t$, we define $w'$ as a *valid WPC* if, after flying to $w'$, the drone has sufficient time to upload the stored data before their deadlines and return to the depot before $t^*$. In particular, we first compute its earliest data uploading time, i.e., $t'' = t + FLT_d(w, w') + FLT_d(w', \mathcal{UP}(w'))$. We say, at current state, $w'$ is valid if $t'$ is before the earliest deadline for its stored data, i.e., $t'' \le \min\{DL_i : T_i \in \mathbf{T}_d, SR_i > 0\}$ and early enough to return in time, i.e., $t'' + FLT_d(\mathcal{UP}(w'), w_{dpt}) \le t^*$. The DFP approach filters out all invalid WPCs in each waypoint selection.

**Fast coverage and reward improvement.** Next, we illustrate the workflow of our DFP approach. Intuitively, we want to finish all released subtasks, then improve the data value as time allows. Thus, we split our flight planning algorithm into two phases: *fast coverage* and *reward improvement*. We aim to finish all subtasks before their deadlines in the first phase. To do this, we maintain a queue of all released subtasks grouped by their deadlines *Que*, sand update it whenever certain subtasks are released or expired. In the algorithm, we use a flag *STUpdate* to track the tasks' update; once

its subtasks are newly released, the *STUpdate* will be set to True during the state transition. We prioritize all groups in the queue by giving higher priority to those with an earlier deadline and then let the drone execute these groups of subtasks following their priorities (lines 1 to 9). To execute each subtask group in the queue $Que[i]$, we get the set of unfinished subtasks, by $\mathbf{T}_{not} = \{T_i, T_i \in Que[i], UR_i = SR_i = 0\}$, and the associated a subset of WPCs that cover those unfinished subtasks; we greedily *choose the next valid waypoint that can maximize the the ratio between the number of unfinished subtasks in Que[i] and the flying time $FLT_d(w, w')$* (line 7). We find the number of $w'$'s covering unfinished subtasks by $CovNum(w') = |\{T_i : T_i \in \mathbf{T}_{not}, V_d(w', T) > 0\}|$. This is repeated until no valid WPC remains or we reach the last group in the queue. Then, if the drone's storage is not empty, we will let the drone upload stored data by flying to the upload point of its current waypoint and continue to cover unfinished subtasks from the first group in the queue (lines 10, 15, and 16). Note that the minimum deadline for drone-stored data impacts the validity of WPCs; after the drone uploads data, the WPCs' validity may change, allowing more valid WPCs to cover other unfinished subtasks.

If no stored data remained, we enter the reward improving phase (line 10) where we improve the data value by having the drone fly lower. We continuously select a waypoint to maximize the reward improvement, i.e., $Reward(w') = \sum_{T_i \in \mathbf{T}_d} \max(0, V_d(w', T_i) - \max(UR_i, SR_i))$. This phase ends when there are no valid WPCs, after which the drone will return to upload data or return to the depot if needed (lines 13 and 15). When subtasks are released (*STUpdate* = True), we update the subtasks queue and restart the fast coverage phase (lines 9, 14, and 16). In our approach, these two phases alternate until the end of the epoch. The computational complexity of our DFP algorithm is $O(\frac{t^* - t_0}{T_{loi}}|\mathbf{W}_d||\mathbf{T}|)$, where the $\frac{t^* - t_0}{T_{loi}}$ indicates the times of waypoint selections when drone stays static. Algs. 4 and 5 in Appendix A shows DFP's detailed pseudo-code.

## 6 EXPERIMENTAL EVALUATION

**Prototype Implementation.** Due to the limited Rx burn opportunities in real forests, we created a lab-based testbed and prototype system (see Fig. 5(a)) to evaluate the implementation and development challenges in DOME. A mock-up burn site ($6 \times 8$ ft) was partitioned into $3 \times 4$ grids. We used a DJI Tello drone equipped with a top-down RBG camera for data collection and the Tello SDK interface for flight control. Commands were provided to guide the drone to follow a waypoint sequence to capture/transmit images at each waypoint. Data was exchanged with a drone controller (a laptop) through a WiFi UDP port. Time and location-stamped images are returned to the ground controller (GC) through the local network, and stored in an edge database. Then, in the GC, the data analyzer processes the received image data to detect and locate fires. All detected fires, with their detection time and location, are stored in the database and reported to the task generator, which updates the states of grid cells and generates corresponding tasks at the start of each epoch (set to 3 min). All generated tasks are submitted to the flight planner to produce the drone flight in this epoch. This flight plan is then sent to the drone controller. This system also maintains a dashboard showing all detected events and the fire spread predictions to visualize the fire status.

### 6.1 Simulation setup

We simulate the Rx burns using the burn plans developed for Blodgett Forest Research Station; we evaluate DOME at three burn sites with different areas/shapes, as shown in Fig. 5(b). In our simulation, a sequence of fire stripes is added progressively, as shown in two sub-figures in Fig. 5(b). We run each Rx burn simulation for 80 mins, with an epoch of 20 mins. At the beginning of each epoch, we input the simulated fire status into the task generator to produce tasks for the $\mathbb{BM}$, $\mathbb{FI}$, and $\mathbb{FT}$ missions, using our proposed rules in §3 under time-driven mode. We perform task allocation and single-drone flight planning to schedule the flights of multiple drones in this epoch to fulfill these generated tasks. We get the missions' PPM requirements from Table 1. In our simulation, we utilize two types of drones; the type 1 drone includes DJI Zenmuse XT2's RGB and thermal cameras, and the type 2 drone includes a DJI Air 2S's RGB camera. We assume drones capture data (RGB/thermal images) when they arrive at a waypoint in their flight path, upload data when connected with the GC, and otherwise store the data onboard. We vary several simulation parameters, including the wind speed ($5 - 25$ mph) and the number of drones ($4 - 16$ drones). Table 2 in the Appendix lists more simulation parameters. Our study records the following performance metrics: (i) *Total Reward*, as defined in the MFP's objective function in Eq. (4a); (ii) *Total Missing subtasks*, which is the number of unfinished subtasks after an epoch; and (iii) *Running time*, which measures the computation time of algorithms. We repeat each experiment 10 times and report the average results with 95% confidence intervals.

We compared DOME techniques with state-of-the-art baseline approaches: lightweight heuristics suitable for real-time planning. For the task allocation step, we compared DOME's UTA (Utilization-based Task Allocation) approach with the Voronoi decomposition algorithm (VD) [23], a popular spatial-based area partition approach used in multi-agent task allocation problems. UTA was also compared with the area-based heuristics task allocation algorithm (ATA), where task execution time relies on the ratio between the total area of tasks and the drone's coverage area. For the flight planning step, we compared DOME's algorithms (DFP) with the Nearest-Neighbor approach (NN) [20], a classical greedy algorithm for vehicle routing problems. We also compared it with an extended version of the Earliest-Deadline-First baseline algorithm [39], a traditional heuristic approach for task scheduling problems; in particular, we added a spatial component to create the Deadline-based Nearest Neighbor (DNN) baseline. We also developed two additional heuristics, a Reward Maximization algorithm (RM), which selects waypoints to maximize the task reward improvement, and Deadline-based Reward Maximization (DRM) to improve the reward of tasks with the earliest deadlines. For a fair comparison, we incorporated the validity checking procedure (in §5.2) into all flight planning baseline algorithms to route the drone to update data in time. We evaluate different combinations of task allocation and flight planning algorithms to solve the MFP problem.

### 6.2 Experimental Results

**Comparative Performance.** Fig 6 compares the performance of the proposed UTA-DFP algorithm with baseline algorithms in burn site ② (see results in burn site ③ in Fig. 10 in the Appendix). We provide sample results from simulations with six drones (three of
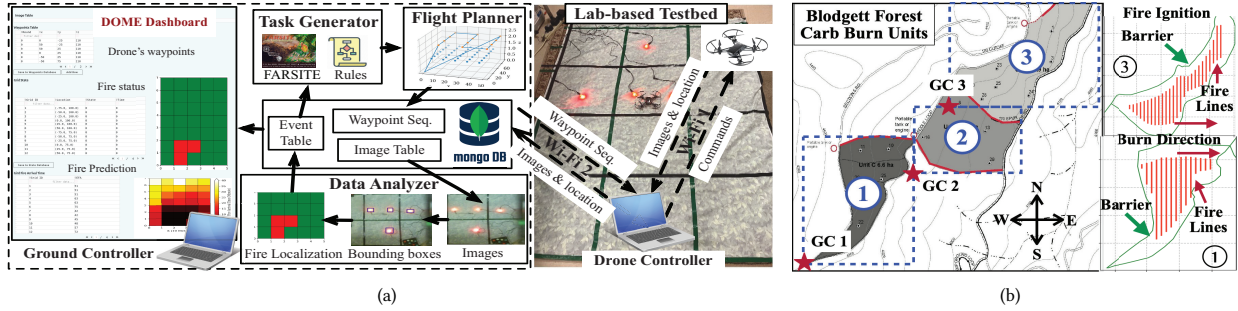
Figure 5: (a) Implementation of our DOME system (b) Illustration of three burn sites and fire ignition strategy.

each type) under 10 mph wind condition over four epochs, each lasting 20 mins. At the beginning of each epoch, we use the defined rules in §3 to generate tasks based on the simulated fire spread status. The number of generated tasks for diverse missions is shown in Fig.11 in the appendix. We specifically illustrate the comparisons with the combinations of VD and RM algorithms and omit the others as they are the best-performance baseline algorithms for task allocation and flight planning, respectively. Fig 6 shows that *our UTA-DFP algorithm always leads to higher total reward and fewer missing subtasks, compared with other baseline algorithms in diverse scenarios.* In particular, our UTA-DFP algorithm achieves 1.7 times gain on total reward, 99% fewer missing subtasks compared with the VD-RM algorithm during 60-80 mins. We also can observe that UTA-DFP and UTA-RM algorithms can consistently provide a relatively steady performance than using VD for task allocation, whose performance drops as the number of tasks increases.

**Task allocation algorithms.** Next, we evaluate our approach for task allocation in Fig. 7. Figs. 7(a) and 7(b) compare our proposed UTA algorithm against other baseline algorithms, generally using the DFP algorithm for flight planning. The results show that *our proposed UTA-DFP algorithm consistently outperforms the other two baseline algorithms* by achieving 0.46% higher total reward, while missing 91% fewer subtasks, compared with the ATA-DFP algorithm during the 60-80 min interval. Also, VD's performance drops during the 40-80 min interval, when more high-frequency tasks for $\mathbb{FT}$ and $\mathbb{FI}$ are generated. This reveals that UTA can handle task heterogeneity better than distance-based approaches.
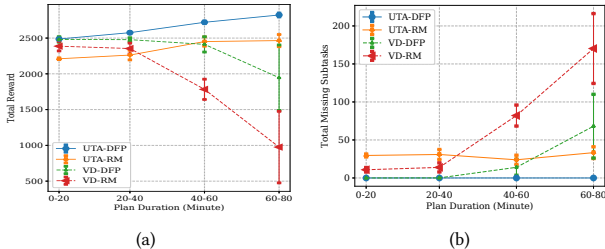


Figure 6: Performance of our UTA-DFP algorithm at Burn sites ②, (a) gives total reward and (b) gives total missing subtasks.

**Flight planning algorithms.** Next, we compare our proposed DFP algorithm with our baseline algorithms when using the UTA algorithm for task allocation in Figs. 7(c) and 7(d). The results demonstrate *our DFP flight planning algorithm always leads to higher overall reward and fewer missing subtasks*, which achieves 0.16 times
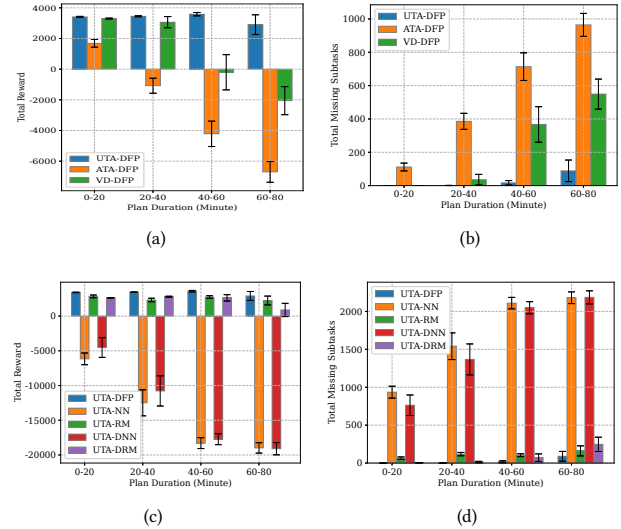


Figure 7: Performance of (a), (b) UTA and (c), (d) FDP algorithms. (a), (c) give the total reward, (b), (d) give the missing subtasks.

gain and 99.5% fewer compared with UTA-NN algorithm. In baseline algorithms, the distance-based baseline algorithms (NN and DNN) always perform worse than the reward-based algorithms (RM and DRM). The results show that DRM and DNN lead to fewer missing subtasks in the first three duration. This observation reveals that considering the temporal factor–deadlines can help improve task accomplishment, but this factor helps less when the number of tasks is much more than drones can complete (e.g., during 60-80 min) Figs. 7(c) and 7(d) show that our DFP algorithm can provide a higher total reward than DRM while they delivering the same number of missing subtasks (in 0-40 min). This confirms that DFP can improve both task accomplishment and data quality.

**Scalability of our proposed algorithm.** Next, we consider a larger MFP problem with a different number of drones (from 8 to 16) and tasks (from 4000 to 6000). Fig. 8(a) and (Fig. 12(a) in Appendix) present the performance of these algorithms in burn site 1 during 60-80 min under 10 mph wind under a diverse number of drones. In all cases, the number of type 1 equals that of type 2 drones. Both figures reveal that our UTA-DFP algorithm always produces the highest total reward and fewer missing subtasks, which achieve 0.43 times gain on total reward and 99.5% fewer missing subtasks compared with VD-RM when drones' number equals 16. Also, the performance of all algorithms becomes better as the number of drones increases. Fig. 8(b) and (Fig. 12(b) in Appendix) show our

algorithms' performance in scenarios with various wind speeds. We give this sample results from simulations with 6 drones in burn site ① during 40-60 min. From the statistics in Fig.11(a) in the appendix, we can see that stronger wind brings out more tasks for 𝔽𝕋 and 𝔽𝕀. As the wind becomes stronger, we can see the superior performance of our UTA-DFP algorithm compared to the baseline ones, which delivers 0.9 times gain on the total reward and about 99% fewer missing events compared with the VD-RM algorithm when wind speed is 5 mph. Fig. 13 show the running time of our UTA-DFP algorithm at burn sites ① and ② . The results show that, in scenarios of about 6000 tasks (during 60-80 min at burn site ①), it can terminate in about 7 seconds, which is negligible to the epoch length, which is 20 min. It depicts that our UTA-DFP algorithm reacts fast even with a large problem size.
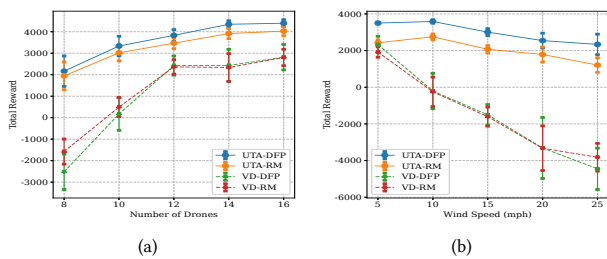


**Figure 8: Performance of the UTA-DFP algorithm with (a) the diverse number of drones and (b) diverse wind speed.**

## 7 CONCLUSION AND FUTURE WORK

The DOME platform was designed to support drone-based monitoring in emergent events, with Rx fires as a driving use case. DOME includes a task generation procedure that combines a physical model with a logic-based approach to generate detailed spatial-temporal sensing requirements. We formulate and solve the multi-drone flight planning problem with heterogeneous drones under disconnected networks, and address the data quality/coverage tradeoff. Our evaluations show the superior performance of our proposed algorithms for various scenarios. In future work, we will explore reinforcement learning and multi-network integration to improve flight planning, onboard image processing and data transmission issues for better timeliness and accuracy. DOME will also be extended for other mission-critical scenarios, e.g., floods and earthquakes, by adjusting monitoring requirements and redefining task generation rules. Integrated mobile sensing (land and aerial) platforms will enable community resilience worldwide for years to come.

## REFERENCES

[1] 2020. PyKnow: Expert Systems for Python. https://github.com/buguroo/pyknow.
[2] Constructions Aeronautiques et al. 1998. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.* (1998).
[3] M. A. Akhloufi, A. Couturier, and N. Castro. 2021. Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. *Drones* (2021).
[4] Enrique Alba and Rafael Martí. 2006. *Metaheuristic procedures for training neural networks.* Vol. 35. Springer Science & Business Media.
[5] H. Azpúrua et al. 2018. Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys. *Robotica* 36, 8 (2018).
[6] J. A Baijnath-Rodino, S. Li, et al. 2022. Historical seasonal changes in prescribed burn windows in California. *Science of the total environment* (2022).
[7] R. W. Beard, T. W. McLain, et al. 2002. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Trans. on Robotics and Auto* (2002).
[8] T. M. Cabreira, L. B. Brisolara, and Paulo R Jr. Ferreira. 2019. Survey on coverage path planning with unmanned aerial vehicles. *Drones* (2019).
[9] N. Chiaraviglio, T. Artés, et al. 2016. Automatic fire perimeter determination using MODIS hotspots information. In *12th Int. Conf. on e-Science.*
[10] Howie Choset. 2001. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence* 31 (2001), 113–126.
[11] IBM ILOG Cplex. 2009. V12. 1: User's Manual for CPLEX. *Intel.* (2009).
[12] Matthew Crosby et al. 2013. Automated agent decomposition for classical planning. In *Intel. Conference on Automated Planning and Scheduling.*
[13] Carmelo D. F. et al. 2016. Coverage path planning for UAVs photogrammetry with energy and resolution constraints. *Journal of Intel. & Robotic Systems* (2016).
[14] Caroline Maul de A. Lima et al. 2018. Performance Evaluation of 802.11 IoT Devices for Data Collection in the Forest with Drones. In *Global Comms. Conf.*
[15] V. R. Desaraju et al. 2011. Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In *IEEE ICRA.*
[16] DRI. 2021. Thermal Infrared Camera Ratings. https://tinyurl.com/3wex9uzu.
[17] Yeonju Eun et al. 2009. Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. *Journal of aircraft* (2009).
[18] M. A Finney. 1998. *FARSITE, Fire Area Simulator–model development and evaluation.* US Department of Agriculture, Forest Service.
[19] FR1. 2014. 5 Drone Technologies for Firefighting. https://tinyurl.com/y4ldl8mf.
[20] Paulo M. França et al. 1995. The m-Traveling Salesman Problem with Minmax Objective. *Transp. Sci.* 29 (1995).
[21] K. A. Ghamry and Y. Zhang. 2016. Cooperative control of multiple UAVs for forest fire monitoring and detection. In *MESA.*
[22] A. Gunawan, H. C. Lau, et al. 2016. Orienteering Problem: A survey of recent variants, solution approaches and applications. *EJOR* (2016).
[23] J. Kim and H. I. Son. 2020. A Voronoi Diagram-Based Workspace Partition for Weak Cooperation of Multi-Robot System in Orchard. *IEEE Access* 8 (2020).
[24] Gilbert Laporte and Frédéric Semet. 2002. Classical heuristics for the capacitated VRP. In *The vehicle routing problem.* SIAM, 109–128.
[25] S. Leary et al. 2011. Constrained UAV mission planning: A comparison of approaches. *Proceedings of the IEEE Intl. Conf. on Computer Vision* (2011).
[26] Leonardo DRS, Electro-Optical & Infrared Systems. [n.d.]. How To Assess Thermal Camera Range For Site Design, White Paper. shorturl.at/osyHT.
[27] F. Liu, T. Y. Fan, et al. 2021. DragonFly: Drone-Assisted High-Rise Monitoring for Fire Safety. In *Int. Symposium on Reliable Distributed Systems.*
[28] LLC. 2012. North Schell Escaped RX . https://tinyurl.com/bdddnksc.
[29] T. W. McLain et al. 2005. Coordination variables, coordination functions, and cooperative-timing missions. *Journal of Guidance, Control, and Dynamics* (2005).
[30] Ayan Mukhopadhyay et al. 2019. An online decision-theoretic pipeline for responder dispatch. In *ACM/IEEE Intel. Conference on Cyber-Physical Systems.*
[31] Ali Nadi and Ali Edrisi. 2017. Adaptive multi-agent relief assessment and emergency response. *Intel. journal of disaster risk reduction* 24 (2017).
[32] Tobias Nägeli et al. 2017. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)* (2017).
[33] F. Nex and F. Remondino. 2019. Preface: Latest Developments, Methodologies, and Applications Based on UAV Platforms. *Drones* 3, 1 (2019).
[34] Ioannis K Nikolos et al. 2007. UAV path planning using evolutionary algorithms. *Innovations in intelligent machines-1* (2007).
[35] H.X. Pham et al. 2017. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In *Intel. Robots and Systems.*
[36] C. Reardon and J. Fink. 2016. Air-ground robot team surveillance of complex 3D environments. *Intl. Symp. on Safety, Security and Rescue Robotics* (2016).
[37] S. A. Sadat, J. Wawerla, and R. T. Vaughan. 2014. Recursive non-uniform coverage of unknown terrains for uavs. In *Conf. on Intel. Robots and Systems.*
[38] Q. Song, Y. Zeng, et al. 2021. A survey of prototype and experiment for UAV Communications. *Science China Information Sciences* (2021).
[39] J. A. Stankovic, M. Spuri, et al. 1998. *Deadline scheduling for real-time systems: EDF and related algorithms.* Springer Science & Business Media.
[40] Alejandro Torreno et al. 2017. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)* 50 (2017).
[41] Zhiyong Wang et al. 2016. Multi-agent based path planning for first responders among moving obstacles. *Computers, Environment and Urban Systems* 56 (2016).
[42] G. Yang, X. Lin, et al. 2018. A telecom perspective on the internet of drones: From LTE-advanced to 5G. *arXiv preprint* (2018).

# A ADDITIONAL PSEUDO-CODE

**Algorithm 2** Utilization-based task allocation (UTA)

**Input:** Tasks $\mathbf{T}$, Drones $D$, Burn site area, square size for task clustering $\Gamma$
**Output:** Task allocation of $\mathbf{T}^* = \{\mathbf{T}_1, \mathbf{T}_2, \ldots \mathbf{T}_D\}$

1: $\mathbf{C} \leftarrow \emptyset; \mathbf{T}_d \leftarrow \emptyset$ for all $d = [1, D]$;
   // Task clustering: get the set of task clusters $\mathbf{C}$
2: **for** $square, j \leftarrow SplitByAreaAndMission(burnsite, \Gamma)$ **do**
3:    $\mathbf{C}_{square,j} \leftarrow \mathbf{T}.filter(square, j)$ // Tasks in $square$ for mission $j$.
4:    $\mathbf{C} \leftarrow \mathbf{C} \cup \mathbf{C}_{square,j}$
   // Assign each drone an initial cluster
5: **for** $D_i \leftarrow D.groupby(\text{"sensor"})$ **do**
6:    $\mathbf{C}_{tmp} \leftarrow \emptyset; \mathbf{C}_{exec} \leftarrow ChooseExecutableTaskCluster(D_i, \mathbf{C})$
7:    **for** $n \in [1, |D_i|]$ **do** // Get $|D_i|$ clusters far from each other
8:      $C^* \leftarrow \arg\max_{C' \in \mathbf{C}_{exec}} \sum_{C'' \in \mathbf{C}_{tmp}} Distance(C', C'')$.
9:      $\mathbf{C} \leftarrow \mathbf{C} \setminus \{C^*\}; \mathbf{C}_{tmp}.add(C^*)$
   // Map one cluster to one drone by distance to GC and $rng_d$
10:    Sort $D_i$ and $\mathbf{C}_{tmp}$ by $rng_d$ and distance to GC; Add $\mathbf{C}_{tmp}[n]$ to $\mathbf{T}_{D_i[n]}$
11: **while** $(\mathbf{C}) \neq \emptyset$ **do** // Iteratively allocate task cluster
12:    **for** $d \in [1, D]$ **do** // Compute minimum $\mathbb{U}_d$ by adding one cluster
13:      $Min_d \leftarrow \min\{\mathbb{U}_d(\mathbf{T}_d \cup C') : C' \in \mathbf{C}\}$
       $C_d^* \leftarrow \arg\min_{C' \in \mathbf{C}} \mathbb{U}_d(\mathbf{T}_d \cup C')$
14:    $d \leftarrow \arg\min_{d \in [1,D]} Min_d$ // Get the drone of minimum $\mathbb{U}$
15:    $\mathbf{T}_d \leftarrow \mathbf{T}_d \cup \{C_d^*\}, \mathbf{C} \leftarrow \mathbf{C} \setminus \{C_d^*\}$ // Allocate a cluster to a drone
16: **return** $\{\mathbf{T}_1, \mathbf{T}_2, \ldots \mathbf{T}_D\}$

---

**Algorithm 3** Compute time utilization $\mathbb{U}_d(\mathbf{T}')$

**Input:** Drone $d$ and a set of tasks $\mathbf{T}'$, WPCs $\mathbf{W_d}$
**Output:** Drone utilization $\mathbb{U}_d(\mathbf{T}')$

1: **for** $T_i \in \mathbf{T}'$ **do** // Determination WPCs to cover all assigned tasks
2:    **if** $Drone\ d\ cannot\ execute\ T_i\text{'s}\ mission\ m_i$ **then return** $+\infty$;
   // Get the WPC drone $d$ can execute $T_i$ and has maximum coverage.
3:    Get $s' \leftarrow \arg\max_{s \in sen_d} CR_s(H(s))$ with $H(s) = max\{h : PPM_s(h) \geq \min(\mathbf{TH}(m_i, s)), h \in AllHeights(\mathbf{W}_d)\}$
4:    Get $WPC_d(T_i)$, a WPC in $\mathbf{W}_d$ at height $H(s')$ can cover $T_i$
5: **for** $k \in [1, M]$ **do** // Compute $\mathbb{U}_d(\mathbf{T}')$
6:    Get tasks for mission $k$: $\mathbf{T}'_k \leftarrow \{T_i : T_i \in \mathbf{T}', m_i = k\}$
7:    Get WPCs for executing all tasks $\mathbf{T}'_k$: $\mathbf{WPC} \leftarrow \{WPC_d(T_i) : T_i \in \mathbf{T}'_k\}$
8:    $ET_d(\mathbf{T}'_k) \leftarrow 0; w' \leftarrow w_0^d$
9:    **while** $\mathbf{WPC} \neq \emptyset$ **do** // Get traversing time for all selected WPCs
     by Nearest Neighbor approach
10:      Choose the WPC closest to $w'$: $w \leftarrow \arg\min_{w \in \mathbf{WPC}} FLT_d(w', w)$
11:      $ET_d(\mathbf{T}'_k) \leftarrow ET_d(\mathbf{T}'_k) + \min\{FLT_d(w', w) : w \in \mathbf{WPC}\}$
12:      $\mathbf{WPC} \leftarrow \mathbf{WPC} \setminus \{w\}; w' \leftarrow w$
   // Add time for uploading data
13:    $ET_d(\mathbf{T}'_k) \leftarrow ET_d(\mathbf{T}'_k) + \frac{\max(0, Dis(w', g_0) - rng_d)}{spd_d} + T_{loi}$
14: **return** $\sum_{k=1}^{M} ET_d(\mathbf{T}'_k)/p(k)$

---

**Algorithm 4** DOME Flight Planning (DFP)

**Input:** Initial state $(w_0, t_0, \mathbf{S}_0(\mathbf{T}_d))$, ending time $t^*$ and WPCs $\mathbf{W}_d$
**Output:** Waypoint sequence $\mathbf{Q}^d$ of drone $d$ and total reward $\mathbb{R}_d$.

1: $STUpdate \leftarrow False; \langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle \leftarrow State_0, \mathbb{R}_d \leftarrow 0$;
2: Group all subtasks based on their deadlines, sort them from $d_i$ smallest to largest: $Que = [\{T_i : T_i \in \mathbf{T}_d, DL_i = d_1\}, \{T_i : T_i \in \mathbf{T}_d, DL_i = d_2\}\ldots]$.
3: **while** $t \leq t^*$ **do**
4:    **if** $STUpdate$ **then** Update $Que$ for new released subtasks ;
5:    $Seq \leftarrow 0$ // **Enter Fast Coverage phase.**
6:    **while** $Seq < |Que|$ **and** $\neg STUpdate$ **and** $t \leq t^*$ **do**
7:      $\mathbf{T}_{not} \leftarrow \{T_i : T_i \in Que[Seq], UR_i = SR_i = 0\}$// Get Unfinished subtasks in this group
8:      **while** $|\mathbf{T}_{not}| > 0$ **and** $\neg STUpdate$ **and** $t \leq t^*$ **do**
9:        $\mathbf{W}' \leftarrow \{w' : w' \in \mathbf{W}_d, CovNum(w') > 0, \text{VALID}(w')\}$
10:        **if** $|\mathbf{W}'| = 0$ **then break**; // If no valid WPC, next group.
11:        $w^* \leftarrow \arg\max_{w' \in \mathbf{W}'} (CovNum(w')/FLT_d(w, w'))$
         // Maximize the number of covered unfinished subtasks
12:        $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUpdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$;
13:        $\mathbf{Q}^d.add(w'); \mathbb{R}_d \leftarrow \mathbb{R}_d + \mathcal{AR}$
14:      $Seq \leftarrow Seq + 1$ // Go to the next group.
15:    **if** $|\{T_i : T_i \in \mathbf{T}_d, SR_i > 0\}| = 0$ **then** // If no stored data.
16:      **while** $\neg STUpdate$ **and** $t \leq t^*$ **do** // **Start Reward Improvement.**
17:        $\mathbf{W}' \leftarrow \{w' : w' \in \mathbf{W}_d, \text{VALID}(w')\}$ // Check validity.
18:        **if** $|\mathbf{W}'| = 0$ **then break**;
19:        $w^* \leftarrow \arg\max_{w' \in \mathbf{W}'} (Reward(w')/FLT_d(w, w'))$
         // Get next state, action reward and the update flag
20:        $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUpdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$;
21:        $\mathbf{Q}^d.add(w'); \mathbb{R}_d \leftarrow \mathbb{R}_d + \mathcal{AR}$
22:    **if** (not $VALID(\mathcal{UP}(w))$) **then** $w' \leftarrow w_0^d$; // Return back
23:    **else** $w' \leftarrow \mathcal{UP}(w)$ // Upload data to the GC.
24:    $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUpdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$;
25:    $\mathbf{Q}^d.add(w'); \mathbb{R}_d \leftarrow \mathbb{R}_d + \mathcal{AR}$
26: **return** $\mathbf{Q}^d, \mathbb{R}_d$.

---

**Algorithm 5** State transition $\text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$

**Input:** Current state $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle$ and next waypoint $w'$
**Output:** Next state $\langle w', t', \mathbf{S}(\mathbf{T}_d) \rangle$, action reward $\mathcal{AR}$, $STUpdate$

1: $t' \leftarrow t + FLT_d(w, w'); \mathcal{AR} \leftarrow 0; STUpdate \leftarrow False$
2: **for** $T_i \in \mathbf{T}_d$ with $\phi_i > t'$ **do** // Update state of tasks
3:    **if** $t' > DL_i$ **then** // Pass subtasks' deadline
4:      $k \leftarrow \lfloor (t' - DL_i)/p_i \rfloor; \mathcal{AR} \leftarrow \mathcal{AR} - k\beta; STUpdate \leftarrow True$
5:      **if** $UR_i = 0$ **then** $\mathcal{AR} \leftarrow \mathcal{AR} - \beta$;
6:      $UR_i \leftarrow 0; SR_i \leftarrow 0; DL_i \leftarrow DL_i + (k + 1)p_i$
   // Drone connects with the GC, uploads data
7:    **if** $\Lambda_d(w') = 1$ **and** $\max(V_d(w', T_i), SR_i) > UR_i$ **then**
8:      $\mathcal{AR} \leftarrow \mathcal{AR} + \max(V_d(w', T_i), SR_i) - UR_i$;
9:      $UR_i \leftarrow \max(V_d(w', T_i), SR_i); SR_i \leftarrow 0$
   // Drone disconnects with the GC, stores data
10:    **else if** $\Lambda_d(w') = 0$ **and** $V_d(w', T_i) > SR_i$ **then**
11:      $SR_i \leftarrow V_d(w', T_i)$
12: **return** $\langle w', t', \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUpdate$

**Table 2: Simulation Parameters**

| Parameter | Value | Parameter | Value |
|-----------|-------|-----------|-------|
| Simulation time | 80 min | Penalty $\beta$ | 10 |
| Epoch length | 20 min | Grid size | $10 \times 10 m$ |
| Wind direction | $270° (\pm30°)$ | Flying speed | 5 m/s |
| Wind speed | 5-25 $(\pm3)$mph | Loiter time | 2 s |
| Ignition interval | 10 $(\pm3)$ min | Drone number | $4 - 16$ |
| Fire strip gap | 10 $(\pm3)$ m | Data trans. range | 300 or 500 m |
| Drone storage | 32 GB | Data trans. rate | 24 Mbps |
| Burn site sizes | ① $400 \times 500$ m, ② $400 \times 330$ m, and ③ $500 \times 420$ | | |
| Mission $(p, \sigma)$ | $\mathbb{BM}$(10 min, 2), $\mathbb{FI}$(5 min, 1), $\mathbb{FT}$(2.5 min, 3) | | |
| Drones | DJI Zenmuse XT2 (XT2); DJI Air 2S (2S) | | |
| Sensor FOV | XT2: thermal $45° \times 37°$, RGB $57° \times 42°$; 2S: $72° \times 58°$ | | |
| Sensor Res. | XT2: thermal $640 \times 512$, RGB $3840 \times 2160$; 2S: $5472 \times 3078$ | | |

# B EVENT-DRIVEN TASK UPDATE

In practice, once tasks are generated, allocated to drones, and executed; they may need to be modified to handle dynamic events. To enable the event-driven task update, we add rules **RE-1** to **RE-5** as defined in Fig. 9, which updates tasks during epochs. The actions $Delete(m, g, t)$ and $Update(m, g, st, et)$ are used to remove and update the time duration of tasks, respectively. Before the monitoring phase, rule **RE-1** deletes tasks for $\mathbb{FD}$ at cells whose state has been changed from **UK** as a result of newly detected events. Rules **RE-2** and **RE-3** designate how to add/delete tasks for $\mathbb{FI}$ and $\mathbb{FT}$ during monitoring when fires are detected/dissipated. Since fires may deviate from expected propagation plans, we define a Rx fire 'anomaly' when they reach a cell earlier than its predicted EFA, potentially denoting escaped or spot fires. This indicates the need for a revised fire prediction and tasks to be updated correspondingly. We use $Anomaly(t)$ to represent an anomaly occurring at time $t$. Rule **RE-4** recognizes an anomaly if fire arrives at $g$ at time $t$ ahead of its $EFA$, i.e., $t \leq EFA(g) - \delta_{an}$, where $\delta_{an}$ is the anomaly threshold. This triggers the fire predictor to rerun the simulation and report the anomaly. Then, rule **RE-5** updates the time duration of tasks for $\mathbb{BM}$ and $\mathbb{FT}$ based on the updated grid cells' EFA. Rules for the event-driven mode are given the lowest priority so that tasks are generated at the start of the epoch before updates.

---

**Facts** :
- *Monitor*: it is true if we have entered the monitoring phase.
- *Anomaly*$(t)$: an abnormal fire spread is reported at time $t$.

**Actions** :
- $\forall g(GetEFA(g, t))$: run FARSITE to obtain EFA of all grid cells based on fire status at time t .
- $Add(m, g, st, et)$: add a task for mission m $\in \{\mathbb{BM}, \mathbb{FD}, \mathbb{FT}, \mathbb{FI}\}$ at g with st and et as its start and end times.
- $Delete(m, g, t)$: remove the task for mission m at g at time t.
- $Update(m, g, st, et)$: update the start and end times of task at g for mission $m \in \{\mathbb{BM}, \mathbb{FD}, \mathbb{FT}, \mathbb{FI}\}$ to st and et respectively .

**Rules** :
**RE-1** : $\forall g(\neg Monitor \wedge (State(g, t) = UK) \wedge (Fire(g, t) \vee NoFire(g, t)))$ $\Rightarrow Delete(\mathbb{FD}, g, t))$
**RE-2** : $\forall g(Monitor \wedge (State(g, t) = NB) \wedge Fire(g, t) \Rightarrow Delete(\mathbb{FT}, g, t) \wedge Add(\mathbb{FI}, g, t, \sim))$
**RE-3** : $\forall g(Monitor \wedge (State(g, t) = B) \wedge NoFire(g, t) \Rightarrow Delete(\mathbb{FI}, g, t))$
**RE-4** : $\exists g(Monitor \wedge (State(g, t) = NB) \wedge Fire(g, t) \wedge (t \leq EFA(g) - \delta_{an})) \Rightarrow \forall g(GetEFA(g, t)) \wedge Anomaly(t + 1)$
**RE-5** : $\forall g(Anomaly(t) \wedge (State(g, t) = NB) \Rightarrow Update(\mathbb{BM}, g, t, \max\{t, EFA(g) - \delta_{ft}\}) \wedge Update(\mathbb{FT}, g, \max\{t, EFA(g) - \delta_{ft}\}, \sim))$

**Figure 9: Rules for event-driven task update**

# C PPM REQUIREMENT

We get these missions' PPM requirements as shown in Table 1, referring to the pixel density requirements for detection, observation, and recognition in Closed Circuit Television (CCTV) systems [16] and Johnson's criteria

[26] that gives the PPM requirements of thermal images for detection, recognition, and identification an object considering the target dimension (height $\times$ width) of human and fire are $1 \times 0.5m$ and $1m \times 1m$ respectively.
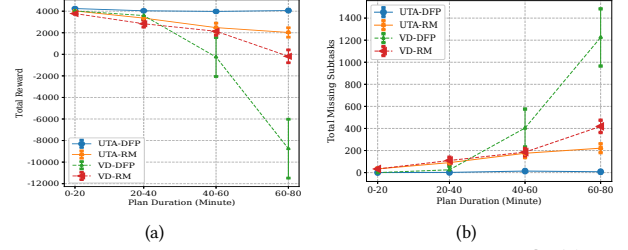


(a)      (b)

**Figure 10: Performance of our UTA-DFP at Burn sites ③. (a) gives the total reward, and (b) the total missing subtasks.**
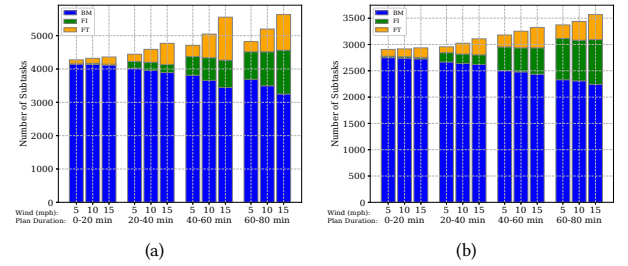


(a)      (b)

**Figure 11: Subtask number in burn sites ① (a),② (b) under diverse wind speeds**
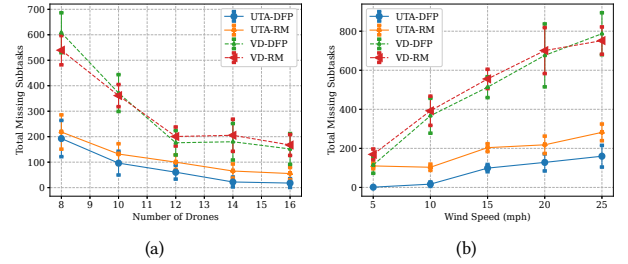


(a)      (b)

**Figure 12: Number of Missing Subtasks of our UTA-DFP under (a) diverse number of drones, (b) diverse wind speed.**
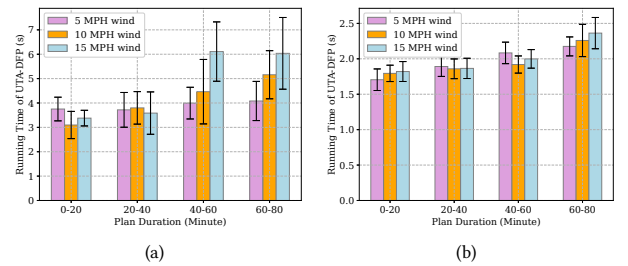


(a)      (b)

**Figure 13: Running time of UTA-DFP at burn sites ① (a) and ② (b)**