DragonFly: Drone-Assisted High-Rise Monitoring for Fire Safety

Fangqi Liu
University of California, Irvine, USA
fangqil2@uci.edu

Tzu-Yi Fan
National Tsing Hua University, Taiwan
joyfan2@gmail.com

Casey Grant
DSRAE, LLC, USA
cgrant.dsrae@gmail.com

Cheng-Hsin Hsu
National Tsing Hua University, Taiwan
chsu@cs.nthu.edu.tw

Nalini Venkatasubramanian
University of California, Irvine, USA
nalini@ics.uci.edu

***Abstract*—In this paper, we propose DragonFly, a drone-based data collection framework to enhance real-time situational awareness in high-rise buildings, focusing specifically on mission-critical high-rise fire scenarios. The goal of our proposed solution is to use multiple drones with visual sensors to collect reliable and timely data for monitoring the exterior of a high-rise building. Drones are especially useful in obtaining data from hard-to-access regions in high-rise fires that are used to monitor fire/smoke that might have propagated to higher floors, detect the presence of humans requiring assistance near windows, and determine window open/close states which can have a significant impact on the speed and direction of fire spread. Given a dynamically evolving set of events and multiple drones, the core challenge addressed is to develop a plan for multiple drones to gather a set of observations that can improve both the coverage (identify more events) and accuracy (obtain fine-grained for improved event detection). We develop a solution for the Multi-drone Waypoint scheduling problem (NP-hard) in two steps: 1) allocation of monitoring tasks (AMT) to individual drones and 2) dynamic waypoint scheduling (DWS) that determines the waypoint sequence for each drone to visit. We evaluate our proposed approach using a simulated high-rise fire scenario with a realistic fire spread model and study the applicability and efficiency of the proposed algorithms compared to baseline techniques. The simulation results demonstrate the superior performance of the proposed AMT-DWS algorithms. DragonFly achieve 33% fewer missing events and up to 39 times gain in accuracy, captured as the minimum weighted AUC (Area Under Curve) as compared to baseline algorithms. DragonFly delivers over 85% missing events and about 1.3 times the minimum weighted AUC in comparison to current approaches.**

***Index Terms*—High-rise fire, drone, observation accuracy, task allocation, waypoint scheduling, optimization.**

I. Introduction

The urban landscape of the future is expected to house over half the world's population; this is incentivizing the growth of mega-cities with high-rise buildings and dense population clusters. Ensuring the safety of humans and other assets in such "vertical" cities, especially during natural/man-made disasters, is challenging – reliable and timely information is required to provide situational awareness in extreme scenarios such as high-rise fires. Today, advances in sensing, mobility, and compute capabilities have made it feasible to create low-cost aerial sensing technologies [1] - drones, UAVs. By serving as "eyes in the sky", data obtained from a carefully coordinated set of drones equipped with sensors have the potential to enable continuous monitoring of mission-critical events. Urban emergencies such as high-rise fires are characterized by dynamic and fast-changing scenarios, where we need to balance rapid identification of emerging events while ensuring the data accuracy is paramount.

In this paper, we address the issue of how to effectively coordinate aerial sensing devices to obtain reliable and timely situational information – we utilize high-rise fires as a driving use-case scenario to study the problem of multi-drone coordination. High-rise buildings have unique properties that make the control of disasters such as fires particularly demanding [2]. Normal response tactics and strategies become significantly less effective with factors such as limited firefighter access, fire spread potential due to dynamic internal air flows, restricted water supplies, wind impact, minimal occupant egress pathways, and special conditions, such as the stack effect. Creating accurate and timely *situational awareness* is critical for fire fighting forces at all levels. Understanding the dynamic hazards and knowing their time-varying states in high-rise buildings are important to both line firefighters and incident commanders because situations can change rapidly and dramatically. Real-time information is therefore crucial.

The use of drones for aerial surveillance and data gathering has great promise as a key tool for urban emergency responses [3]. Modern drones can be equipped with heterogeneous sensors (e.g., optical and hyper-spectral cameras) that are useful for tasks such as scoping the region of the event, heat source detection, and victim localization [4]. In the context of high-rise fires, drones can bring additional values by enabling rapid detection (and mitigation, when possible) of emerging events such as: (i) detecting sudden changes in

This work is supported by the UC Office of the President (#LFR-20-653572), the United States Air Force and DARPA (#FA8750-16-2-0021 and #FA8750-16-C-0011), the National Institute of Standards and Technology (#70NANB17H285), the United States Navy and DARPA (#N66001-15-C-4065, #N66001-15-C-4067, and #N66001-15-C-4070), Ministry of Science and Technology (MOST) of Taiwan (#110-2221-E-007-102 and #107-2221-E-007-091-MY3), a Qualcomm grant, and a NOVATEK Fellowship.

the fresh air feeding fires such as with a window loss during *wind-driven fires*, and (ii) tracking fires involving external building facades or *combustible exterior wall assemblies*. Wind-driven high-rise fire is a special concern to today's fire service. Among the most classic examples of this situation is the 1998 New York City "Vandalia Ave" high-rise fire that resulted in the deaths of three veteran FDNY firefighters trapped in a 10th-floor hallway [5]. Particularly, sudden loss of windows or similar building envelope components are known to create unsurvivable situations [6]. Another major concern for firefighters is the combustible exterior wall assemblies. In parts of the world with significant high-rise constructions in recent years, there is a realization that these buildings are at risk of serious high-rises fire, where the fires may rapidly spread along the exterior surface [7]. There have been multiple such high-rise building fires, e.g., the 2017 Grenfell Tower fire in London with 72 fatalities [8].

In this paper, we take a systematic approach to utilize drones to create rapid and accurate situational awareness. The fundamental methodology of deploying drones has a distinct value in extreme events because of its ability to scale resources and provide flexibility with real-time adjustments. When the fire service is notified of high-rise building fires, the resources arrive over time as the efforts are scaled up. Going forward, this will include deploying drones for fire fighting surveillance and providing live images or videos to fire fighting forces for real-time situational awareness [9]. From a scalability standpoint, the drones are movable units that can canvass the building surfaces over extended time frames to provide real-time surveillance. As drones are added, these resources can be dedicated to specific areas of building facades, or scheduled to provide more frequent updates at the fire scene.

In particular, we propose a multi-drone coordination system, called *DragonFly* which automatically manages drone-based sensing and monitoring at high-rise fire scenes. DragonFly is activated upon the firefighters arrive at fire scenes. It then continuously guides drones to collect sensor data for improving the situational awareness of firefighters. In DragonFly, monitoring tasks are generated and updated by a task generator based on the fire report and the acquired information by drones. A monitoring task specifies the event to be detected, along with some monitoring requirements (e.g., location, significance, and desired monitoring frequency). Together with fire agency partners, we have identified a set of critical events to drive monitoring tasks [10]. DragonFly effectively allocates drones to specific tasks and determines waypoint sequences for drones on-the-fly (within a short decision-making time) to accomplish those tasks. This paper makes the following contributions:

- DragonFly system design for high-rise fire monitoring (Sec. III).
- Formulation of the Multi-Drone Waypoint Scheduling Problem (MWSP) with considerations of the tradeoff between the observation accuracy and monitoring area coverage under heterogeneous tasks (Sec. IV).
- Development of a two-step approach to solve the MWSP (Sec. V).
- Evaluations of our solution using simulations (Sec. VI).

II. Tackling the High-Rise Fire Scene

Fire service, especially for structural fires, is inherently a human-in-the-loop activity coordinated by an *Incident Commander (IC)* at the fire scene, where an *Incident Command Site (ICS)* is established. The IC and a team of analysts digest live data from camera, environments, and other sensors to extract the states of the fire scene and coordinate responses. Given the added challenges of dynamicity in high-rise fires with the possibility of rapid changes due to wind-driven fires and exterior combustibility, and reduced ability in (or lack of) manual observations, the added visual monitoring results from aerial sensing will help drive and navigate the search, rescue, and mitigation missions of the fire service.

Existing work. Drones have frequently been utilized for aerial sensing and surveillance during both non-disaster and disaster times – e.g., building surveillance [11] and post-disaster environment assessment [12]. Drones carrying specific payloads (e.g., fire retardants and extinguishing balls) have been used in wildland fire scenarios [13], where target monitoring areas are selected by fire fighting forces. Recent efforts have also studied the possibility of early localization of building fires using drones [14]. Other related literature has focused on multi-agent waypoint scheduling in a variety of settings, where long-term monitoring with different target perspectives is required [15]. These waypoint scheduling problems are typically cast into combinatorial optimization problems–also referred to as patrolling problems, that seek to minimize the time between two visits of the same waypoint. The literature also includes work on planning for persistent monitoring in 2-D grids under uncertainty [16] and patrolling multiple regions with changing features at different rates [17]. Region partitioning techniques in conjunction with inter-region waypoint scheduling [18] aim to balance visiting workloads across multiple regions. The cooperative approach of waypoint scheduling among multiple drones has been formalized and shown to be NP-hard [19]. A range of heuristic approaches using Mixed Integer Linear Programming (MILP), Markov Decision Process (MDP), and game theory have illustrated the complexity of the problem [20]. The driving use cases for these settings are military command-and-control missions, where drones must move to target areas in the presence of dynamic threats [21] and hostile environments [22]. For example, the techniques for drones to rendezvous at unspecified locations [23] or capture geo-dispersed targets in no-fly zones [24] have both been studied in this setting.

Challenges in high-rise fires. In contrast to the above efforts, the 3-D high-rise fire setting studied in this paper introduces new levels of complexity as follows. First, the environment is dynamic due to the fire spread, human movements, and change of ventilation state (open/broken windows). Drones should continuously monitor the whole fire scene to track time-varying states. Because of the vision obstacles and the limited

sensor ranges, the number of drones might not be sufficient to cover all high-rise building facades at a time. Thus, we need to guide multiple drones to maximize the coverage and minimize the data collection delay. Second, we should also consider the diversity in the monitoring events at the fire scene, when planning drone surveillance. The events, e.g., the presence of fire or human, have their particular properties, w.r.t. dynamics and information significance. Accordingly, the monitoring requirements for them should be differentiated. For example, monitoring the victims near fire sources must be prioritized for emergency rescue, and regions close to fires must be monitored more frequently for detecting fire spread.

Finally, the locations of drones and distances w.r.t. the building affect the monitoring performance and detecting coverage. Recent studies [25]–[27] indicate that object detection accuracy levels with diverse sensors are significantly affected by the distance between the camera and the observation target. With this concern, we consider both *coarse- and fine-grained observations*, where drones capture sensor data (images) at different distances. Specifically, a drone gets a coarse-grained observation when it takes images at a relatively far distance to the building facade, which results in a larger coverage but a lower accuracy level. In contrast, a fine-grained observation is taken at a closer distance that leads to the smaller image coverage but more accurate event detection.

Because of the above concerns, this paper formulates a unique Multi-Drone Waypoint Scheduling Problem (MWSP) for guiding multiple drones to perform monitoring tasks considering the fire-scene dynamics and the heterogeneous emergent events. In this problem, we carefully dictate coarse- and fine-grained observations to exercise the best tradeoff between accuracy and coverage. We solve this problem in two steps: Allocation of Monitoring Tasks (AMT) and Dynamic Waypoint Scheduling (DWS). Different from earlier 2-D task allocation problems [16]–[18], our AMT solution strives to balance the workload of drones while taking into account the event properties (frequency, significance) in 3-D space. For the DWS solution, we define the notion of information accuracy (with decay) to drive the scheduling of drones for coarse- and fine-grained observations when capturing task dynamics. We note that prior studies on motion planning or drone patrolling do not consider such diverse observations and their impacts on the overall situational awareness.

III. The DragonFly Framework

We next provide an overview of the DragonFly framework, which is shown in Fig. 1.

Data receiver and analyzer. Drones with cameras and other sensors are deployed outside a high-rise building to continuously collect data under coarse- and fine-grained observations by adjusting the observation distances to the building facades. More specifically, each drone maintains three data links for sensor data, telemetry (states, such as locations and battery levels), and control commands. All these data are transmitted to ICS, where sensor data are analyzed for detecting *events* to reveal the states of fires, humans, and building ventilation.

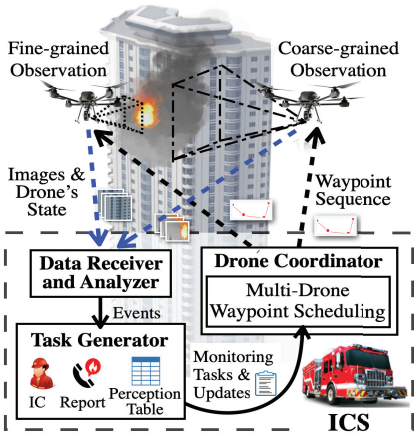


Fig. 1: Overview of DragonFly framework.

Sample events include the presence/absence of fire, the existence of humans, and the open windows or doors. Besides, the events can also be the changes of the fire intensity (temperature and flame size), and the human movement (change of locations or postures). All detected events are stored in a *perception table* with their locations and observation time. This table is initialized with the fire reports (see Table I for an example) and automatically updated to record new detections (see Table II for an example).

Task generator. *Task generator* creates monitoring tasks stored in a task table (see Table III for an example) for guiding drones to collect sensor data. Each monitoring task is associated with a task generation time, a target event, e.g., the presence of a fire or human, a monitoring area, e.g., the area contains Room 601's windows, a *significance level* for firefighting forces, and a *desired frequency* for observations. The significance levels depend on the event to be detected and the distance to the fire. For example, detecting humans near the fire source is more critical. The desired frequency depends on the dynamics of events. For example, monitoring fire intensity should be done more frequently than monitoring window states because fire intensity rapidly changes. Task generator initially generates tasks according to the fire reports, and it may updates tasks with the arrival of newly detected events. For example, if a fire is detected, the task generator should assign a higher significance level to that fire detection task. Such updates are done according to the firefighting domain knowledge [10]. Table III gives the examples of the generated tasks, where Win. in R601 and Win. on F6 denote the windows in Room 601 and all windows on the 6th floor, and the unit of frequency is times per minute.

Drone coordinator. *Drone coordinator* consists of two main components. The *state tracker* records the dynamic drone state, e.g., the location and the power state. Such information allows the DragonFly to track the states of each drones, estimate the observation time of every monitoring area, and predict the corresponding observation accuracy. *Multi-drone waypoint scheduling* block takes states and user-specified parameters to compute the waypoint sequence for

TABLE I: Initial Perception

Event	Location	Time
Fire	R601	10:00

TABLE II: Perception at time 10:05

Event	Location	Time
Fire	R601	10:03
Fire	R602	10:04
Humans	R605	10:02
Open Window	R706	10:05

TABLE III: Task Table

Time	Event	Area	Sig.	Fre.
10:00	Fire	Win. on F6 Win. on F7 Win. on F8	3	0
10:00	Human	Win. on F6 Win. on F7	2	0
10:03	Human Movement	Win. in R605	3	2
10:03	Open Window	Win. in R601 Win. in R602 Win. in R603	2	0.2
10:04	Fire Intensity	Win. in R601 Win. in R602	3	2

each drone. Here, the *waypoints* specify the locations and camera orientations of drones for observing the potential events in coarse- or fine-grained ways. To cope with the system dynamics and unpredictability, we solve the waypoint scheduling problem (called MWSP in this paper) multiple times, each for a fixed time period called *plan duration*. More concretely, a new waypoint sequence would be generated for each drone when the state tracker reports the previous sequence is completed, the monitoring tasks are updated, or drones are added/removed.

In this paper, we assume fire fighting forces have accessed to the high-rise building structure information, such as the floorplans, locations of windows, and mapping between windows and rooms. We believe such assumption is not too strong, as fire fighting forces are part of local governments, while open-data paradigm [28] is getting increasingly popular. We also assume the commands, images, and states exchange between the ICS and drones are over reliable communication networks, such as LTE, 5G, or WiFi [29]. This is also reasonably given the rapid increase in the penetration rate of mobile networks and reliable drone APIs such as DJI Mobile SDK [30]. Last, the design of image analysis algorithms are orthogonal to this paper, any images (or sensor) data analysis algorithms can be adopted by DragonFly and their accuracy levels are given.

IV. Multi-Drone Coordination for High-Rise Fires

A. Monitoring Tasks

We define a *monitoring task* k as a tuple $(g_k, e_k, o_k, \sigma_k, \eta_k)$, where g_k is the task generation time, e_k is the target event, and o_k is the monitoring area, which is a bounding box containing one or multiple windows, doors, or balconies on a building facade. We choose e_k from $\mathbf{E}$, which is the set of all events with $|\mathbf{E}|= E$. Each monitoring task has its own significance σ_k, and desired frequency η_k. We use $\mathbf{K}$ to denote the set of monitoring tasks in the task table at ICS, with $k \in \mathbf{K}$ and $|\mathbf{K}|= K$. We let $\mathbf{M} = \{m_1, ...m_M\}$ indicate the set of all possible monitoring areas at our fire scene. Each monitoring area is defined as $m_i = (\{v_1^i, v_2^i, v_3^i, v_4^i\}, \vec{n}_i)$, $i = 1, .., M$, where v_1^i to v_4^i are the four vertices of the monitoring area and $\vec{n}_i$ is its normal vector. Because the monitoring areas are defined on the building facades, we use the local 2-D coordinate systems of individual building facades to represent the monitoring areas. The conversion between the local 2-D and global 3-D coordinates is straightforward and thus omitted.

We generate a waypoint sequence for each drone to follow and accomplish the monitoring tasks. Each waypoint w is represented as a tuple $(v'_w, \vec{w}'_i)$, where v'_w is the 3-D coordinates and $\vec{w}'_i$ is the orientation vector of the camera (or another sensor). Upon reaching a waypoint, a drone makes an *observation* of the monitoring areas. Here, an observation refers to capturing the sensor data, such as images. By selecting the distance between a waypoint to the building, drones may make coarse- or fine-grained observations to exercise the tradeoff between the accuracy and coverage.

B. Candidate Waypoints

The number of candidate waypoints for the drones to select for performing the monitoring tasks is infinite in theory. To be practical, we discretize the waypoints using a user-specified distance set $\mathbf{D}$ between the waypoints to the corresponding building facades. It is not hard to see that a longer distance $d \in \mathbf{D}$ results in a larger coverage area (f_d^W, f_d^H) of the drone's camera on the building facades, where f_d^W and f_d^H are the width and height of the rectangular area covered in the image captured by a drone hovering at distance d.

Given all monitoring areas $\mathbf{M}$ and possible distances $\mathbf{D}$, our problem is to build a set of promising waypoints $\mathbf{W} = \{w_1, ...w_W\}$, where each $w \in \mathbf{W}$ covers one or more monitoring areas. Here, we say a waypoint w covers an area $m_i = (\{v_1^i, v_2^i, v_3^i, v_4^i\}, \vec{n}_i)$ iff all the four vertices of m_i are within the coverage area of the drone's camera when the drone is at w. In addition to $\mathbf{W}$, we define a coverage matrix $\mathbf{C} = \{C(w, m)\}_{W \times M}$ to map waypoints to monitoring areas, where $C(w, m) = 1$ if the drone at w can cover m, and $C(w, m) = 0$ otherwise.

We build $\mathbf{W}$ as follows. Without loss of generality, we assume (f_d^W, f_d^H) can cover at least an area m entirely; otherwise, we skip the d and m. For each $d \in \mathbf{D}$ and $m \in \mathbf{M}$, there are too many waypoints whose coverages (f_d^W, f_d^H) contain m. For each m, we consider four[1] waypoints, where the coverage of each waypoint shares a corner with the monitoring area m. Next, for each considered waypoint w, we determine a subset of monitoring areas that fall in the coverage (f_d^W, f_d^H). If the subset of monitoring areas w is identical to any known $w' \in \mathbf{W}$, w is no longer considered. Otherwise, we add w to $\mathbf{W}$ and update $\mathbf{C}$ accordingly. We check this to avoid having too many *redundant* waypoints that offer the same coverage of monitoring areas. We return $\mathbf{W}$ and $\mathbf{C}$ after checking all monitoring areas $\mathbf{M}$ and distances $\mathbf{D}$. Last, we use $d(\langle w_i, w_j \rangle)$ to denote the 3-D path length between waypoints $w_i, w_j \in \mathbf{W}$ considering the buildings as obstacles, which can be readily computed using 2-D visibility graph path planning method [31] with elevation difference.

C. Accuracy of Monitoring Tasks

Observation accuracy. Given image (data) analysis algorithms and camera configurations, we deduce the accuracy of the analysis for detecting events on images captured by drones at different distances to building facades. Particularly, we define the accuracy of monitoring task k at distance d –

[1]Denser waypoints can be considered at the cost of higher computational complexity.

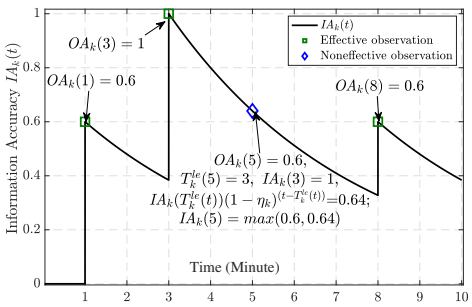


Fig. 2: Sample $IA_k(t)$ with $\eta_k = 0.2$ and $g_k = 0$.

$A(d, e_k)$, with $k \in \mathbf{K}$ and $d \in \mathbf{D}$, to represent the accuracy for detecting event e_k using images collected at distance d.

We use $d_{w_i} \in \mathbf{D}$ to denote the distance from waypoint w_i to the building facade, and write the accuracy of task k at w_i as $A(d_{w_i}, e_k)$. Suppose drones arrive at waypoints along a time sequence $\mathbf{T}^{ar} = [t_1, t_2, ...]$ at waypoints $[w(t_1), w(t_2), ...]$ during monitoring, where $\forall t_i, t_j \in T^{ar}$: $t_i \neq t_j$ if $i \neq j$. We let $OA_k(t)$ be the *observation accuracy* of monitoring task k at time t, which equals the accuracy of data captured by drones at t for detecting event e_k. It is calculated by:

$$OA_k(t) = \begin{cases} A(d_{w(t)}, e_k) \times C(w(t), o_k), & t \in T^{ar} \\ 0, & otherwise. \end{cases} \quad (1)$$

From Eq. (1), we can infer that $OA_k(t) = 0$ if drones do not arrive at any waypoint at t or waypoint $w(t)$ doesn't cover k's monitoring area o_k; otherwise, $OA_k(t) = A(d_{w_i}, e_k)$.

Effective observation and information accuracy. Due to the limited number of drones, monitoring areas are not observed continuously. Therefore, whenever an event state of task $k \in \mathbf{K}$ is queried at t, ICS returns the result of a recent observation of k which is referred to as the *last effective observation*. To measure the accuracy of the queried results, we define the *information accuracy* of task $k \in \mathbf{K}$ at time t as $IA_k(t)$, which equals to the estimated probability that the analysis result of the k's last effective observation is the same as the practical current state of k at t.

We set $IA_k(g_k) = 0$, and assume the degrading of $IA_k(t)$ follows a geometric distribution with a parameter η_k, unless ICS gets an effective observation of task k. We consider an observation of task k at t is an effective observation iff $OA_k(t) > IA_k(T_k^{le}(t))(1-\eta_k)^{(t-T_k^{le}(t))}$, where $T_k^{le}(t)$ denotes the time of the last effective observation of task k at time t, with $T_k^{le}(g_k) = g_k$.

For simplicity, we rule that whenever ICS receives an observation, it updates its perception table following the new observation if it is an effective observation and ignores it otherwise. The information accuracy $IA_k(t)$ is defined as follows, whose sample dynamics is shown in Fig. 2:

$$IA_k(t) = \begin{cases} 0, & t = g_k; \\ \max\{IA_k(T_k^{le}(t))(1-\eta_k)^{(t-T_k^{le}(t))}, & \\ OA_k(t)\}, & t > g_k. \end{cases} \quad (2)$$

We define the *Area Under Curve (AUC)* of monitoring task k at time t' as: $\int_{t=g_k}^{t'} IA_k(t)dt$ to quantify the overall information accuracy of k during time $[g_k, t']$.

D. Formulation of the Multi-Drone Waypoint Scheduling Problem (MWSP)

We next formulate the Multi-Drone Waypoint Scheduling Problem (MWSP) for scheduling N drones to fulfill monitoring tasks $\mathbf{K}$ by visiting a set of waypoints $\mathbf{W}$ during time $[t_0, t_0+T]$, where t_0 is the current scheduling time and T is the plan duration. The drones depart from their initial waypoints $[w_{in(1)}, \ldots, w_{in(N)}]$ at time t_0 and are required to return to a depot w_0 by $t_0 + T$.

We define a boolean matrix $\mathbf{X} = [x_{i,j}^{n,s}]$, where $w_i, w_j \in \mathbf{W}$, $n \in [1, N]$ and $s \in [1, S]$, to represent the waypoint sequences of all drones. In particular, $x_{i,j}^{n,s} = 1$ indicates that drone n flies from waypoint w_i to its s-th waypoint w_j, and $x_{i,j}^{n,s} = 0$ otherwise. Here, S represents the maximal length of waypoint sequences of all drones.

Given $\mathbf{X}$, we write the arrival times of drone n at waypoints as $\mathbf{T}_n^{ar}(\mathbf{X}) = [T_n^{ar}(1), ...T_n^{ar}(S)]$, where $T_n^{ar}(s)$ denotes the arrival time of drone n at its s-th waypoint, with $s \in [1, S]$ and $T_n^{ar}(s) \in [t_0, t_0 + T]$. It is computed by:

$$T_n^{ar}(s) = t_0 + \sum_{s'=1}^{s} \sum_{w_i, w_j \in \mathbf{W}} (\frac{d(\langle w_i, w_j \rangle)}{R_{fly}} + T_{loi}) x_{i,j}^{n,s'}, \quad (3)$$

In this equation, R_{fly} is the drones' flying speed, and T_{loi} is the loiter time at each waypoint.

Then, we can derive $OA_k^n(T_n^{ar}(s), \mathbf{X})$, which indicates the observation accuracy of task k when drone n arrives at each waypoint by:

$$OA_k^n(T_n^{ar}(s), \mathbf{X}) = \sum_{w_i, w_j \in \mathbf{W}} C(w_j, o_k) A(d_{w_j}, e_k) x_{i,j}^{n,s}. \quad (4)$$

We also let $OA_k^n(t, \mathbf{X}) = 0$, $\forall t \notin \mathbf{T}_n^{ar}(\mathbf{X})$. Thus, we can redefine $OA_k(t)$ of Eq. (1) by considering the possibility that multiple drones cover a area simultaneously. The new observation arraucy is:

$$OA_k(t, \mathbf{X}) = \max_{n \in [1,N]} \{OA_k^n(t, \mathbf{X})\} \quad (5)$$

Given $\mathbf{X}$, the information accuracy of each task $k \in \mathbf{K}$ during $t \in [t_0, t_0 + T]$ can be written as:

$$IA_k(t, \mathbf{X}) = \begin{cases} IA_k(T_k^{le}(t_0))(1-\eta_k)^{(t_0 - T_k^{le}(t_0))}, & t = t_0; \\ \max_{n \in [1,N]}\{IA_k(T_k^{le}(t))(1-\eta_k)^{(t-T_k^{le}(t))}, & \\ OA_k^n(t, \mathbf{X})\}, & t > t_0. \end{cases} \quad (6)$$

Considering that tasks $\mathbf{K}$ may be generated or fulfilled before the scheduling time t_0, MWSP aims to schedule multiple drones during $[t_0, t_0 + T]$ to improve the AUC of all tasks within $t \in [g_k, t_0 + T]$. This can be written as:

$$\int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{X})dt = \int_{t=g_k}^{T_k^{le}(t_0)} IA_k(t)dt + \int_{t=T_k^{le}(t_0)}^{t_0+T} IA_k(t, \mathbf{X})dt. \quad (7)$$

Here, we assume the state tracker in the drone coordinator who is continuously tracking the monitoring history provides $\int_{t=g_k}^{T_k^{le}(t_0)} IA_k(t)dt$, $T_k^{le}(t_0)$, and $R_k(T_k^{le}(t_0))$ of all monitoring task $\mathbf{K}$ at each scheduling time t_0. In this way, MWSP takes the various completion status of tasks at t_0 into account when performing waypoint scheduling.

With above notations, we formulate the MWSP as follows:

$$\max \min_{k \in \mathbf{K}} \{\frac{1}{\sigma_k} \int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{X})dt \frac{1}{t_0 + T - g_k}\} \quad (8a)$$

$$\text{s.t.} \sum_{s=1}^{S}(\frac{d(\langle w_i, w_j\rangle)}{R_{fly}} + T_{loi}) \sum_{w_i\in\mathbf{W}} \sum_{w_j\in\mathbf{W}} x_{i,j}^{n,s} \leq T; \quad (8b)$$

$$\sum_{w_i\in\mathbf{W}} x_{i,h}^{n,s} \times \sum_{w_i\in\mathbf{W}} x_{i,h}^{n',s'} \neq 1,$$

$$\forall w_h \in \mathbf{W} \setminus \{w_0\},\ T_n^{ar}(s) = T_{n'}^{ar}(s'),\ n \neq n'; \quad (8c)$$

$$\sum_{w_i\in\mathbf{W}} \sum_{w_j\in\mathbf{W}} x_{i,j}^{n,s} = 1; \quad (8d)$$

$$\sum_{w_j\in\mathbf{W}} x_{in(n),j}^{n,1} = \sum_{w_i\in\mathbf{W}} x_{i,0}^{n,S} = 1; \quad (8e)$$

$$\sum_{w_i\in\mathbf{W}} x_{i,0}^{n,s} \leq x_{0,0}^{n,s+1}; \quad (8f)$$

$$\sum_{w_i\in\mathbf{W}} x_{i,j}^{n,s''} = \sum_{w_z\in\mathbf{W}} x_{j,z}^{n,s''+1},\ \forall s'' \in [1, S-1]; \quad (8g)$$

$$x_{i,j}^{n,s} \in \{0,1\}; \quad (8h)$$

$$\forall w_i, w_j \in \mathbf{W},\ s, s' \in [1, S],\ n, n' \in [1, N].$$

The objective function in Eq. (8a) maximizes the minimal weighted information accuracy across all monitoring tasks, where $\frac{1}{t_0+T-g_k}$ is a normalization factor. The intuition of introducing weight $\frac{1}{\sigma_k}$ here is to provide higher information accuracy to more significant tasks. The constraint in Eq. (8b) ensures that the total time spent by each drone is within the plan duration T. Eq. (8c) guarantees that at most one drone reaches a waypoint at a specific time, which avoids collisions and interference among drones. A drone visits one waypoint in each step, which is captured by the constraint in Eq. (8d). The constraints in Eqs. (8e) and (8f) set the initial and final waypoints for all drones. Eq. (8g) ensures the connectivity of the generated waypoint sequences. Last, Eq. (8h) specifies that $x_{i,j}^{n,s}$ is a boolean value. MWSP is NP-hard, which can be proven through reducing the Traveling Salesman Problem (TSP) [32] to a special case of MWSP, which has a single drone, only one kind of task with $\eta_k = 0$, a constant observation distance with accuracy 1, $t_0 = g_k$ for all tasks and plan duration $T = \infty$.

The max-min objective function in Eq. (8a) strives for fairness, as additional resources are always allocated to the task with the lowest information accuracy. Nonetheless, alternative objective functions are possible, such as

$$\max \sum_{k\in\mathbf{K}} \int_{t=g_k}^{t_0+T} \sigma_k IA_k(t, \mathbf{X}) dt \frac{1}{t_0 + T - g_k}, \quad (9)$$

if the average information accuracy is more important than the max-min fairness. If not otherwise specified, we adopt the max-min objective function throughout the paper because the worst-case scenario carries much higher weight in high-rise fires.

V. Proposed Algorithms for MWSP

Given the real-time nature of the MWSP problem and associated complexity (NP-hard), we propose to solve this problem by two steps, in each of which a sub-problem is solved heuristically. The first step solves the *Allocation of Monitoring Tasks (AMT)* problem, which allocates a set of monitoring tasks to each drone. The second step solves the *Dynamic Waypoint Scheduling (DWS)* problem, which determines a waypoint sequence for each drone to visit. Fig. 3 illustrates the workflow of our MWSP solution, which is detailed in the following.

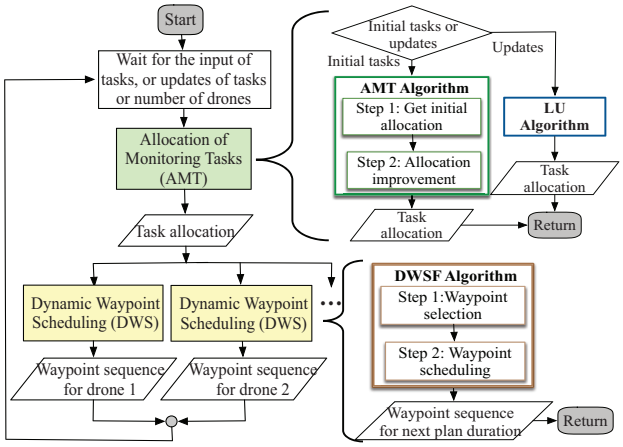


Fig. 3: Workflow of our proposed algorithms.

A. Allocation of Monitoring Tasks: AMT

Given monitoring tasks $\mathbf{K}$, we first derive the set of monitoring areas: $\mathbf{M^K}$ where $\mathbf{M^K} \subseteq \mathbf{M}$ and $o_k \in \mathbf{M^K}\ \forall k \in \mathbf{K}$. The AMT problem spatially allocates $\mathbf{M^K}$ into N disjoint subsets $\mathbf{M'} = \{\mathbf{M'}_1, ..., \mathbf{M'}_N\}$, with $\mathbf{M^K} = \bigcup_{n=1}^{N} \mathbf{M'}_n$. Given $\mathbf{M'}$, we allocate all monitoring tasks $\mathbf{K}$ into N disjoint subsets represented by $\mathbf{K'} = \{\mathbf{K'}_1, ...\mathbf{K'}_N\}$ with $\mathbf{K} = \bigcup_{n=1}^{N} \mathbf{K'}_n$. Here $\mathbf{K'}_n$ is the set of tasks assigned to drone n which are within the monitoring area $\mathbf{M'}_n$, i.e., $\mathbf{K'}_n = \{k \mid k \in \mathbf{K}, o_k \in \mathbf{M'}_n\}$.

We employ the graph structure to represent the spatial correlations among monitoring areas. More specifically, we define a complete graph $\mathbf{G} = (\mathbf{M^K}, \mathbf{E})$, where the nodes indicate the monitoring areas $\mathbf{M^K}$, and edges $\mathbf{E} = \{\langle m_i, m_j\rangle \mid m_i, m_j \in \mathbf{M^K}\}$ are the pairwise links between any two monitoring areas. We use the center of area $m \in \mathbf{M^K}$ to represent each node location, and $d(\langle m_i.m_j\rangle)$ to denote the edge length which equals to the 3-D path length between two areas considering building as obstacles[2].

Based on MWSP's objective function in Eq. (8a), we come up with two intuitions. First, the optimal task allocation minimizes the overall (maximum) time consumption of drones for traveling among areas to accomplish all their allocated tasks. Second, it offers more observation opportunities to the areas with monitoring tasks having higher significance or frequency. With the above intuitions, we define the *desired number of observations* of area $m_i \in \mathbf{M^K}$ throughout plan duration T as $B(m_i)$, which is a function of task significance and frequency. More precisely, we write it as:

$$B(m_i) = \begin{cases} 1, \ \max_{k\in\mathbf{K}, o_k=m_i}\{\eta_k\} = 0, \exists k \in \mathbf{K} : o_k = m_i; \\ \lceil T \times \max_{k\in\mathbf{K}, o_k=m_i}\{\sigma_k \times \eta_k\}\rceil, \qquad otherwise. \end{cases} \quad (10)$$

[2]Distance $d(\langle m_i.m_j\rangle)$ between two areas is calculated in the same way as that between two waypoints defined in Sec. IV-A.

Then, we define the *expected time consumption* for monitoring set $\mathbf{M}'_n \in \mathbf{M}'$ as the time drone n spends for observing all monitoring tasks for desired numbers of times as:

$$ET(\mathbf{M}'_n) = \sum_{m_i \in \mathbf{M}'_n} B(m_i)(\frac{2 \times d(\langle m_i, \hat{m}_n \rangle)}{R_{fly}} + T_{loi}), \quad (11)$$

where $\frac{2 \times d(\langle m_i, \hat{m}_n \rangle)}{R_{fly}}$ is the round trip time between $\hat{m}_n$ and m_i, T_{loi} is the loitering time of drones at each monitoring area, and $\hat{m}_n$ is the center node of $\mathbf{M}'_n$ with

$$\hat{m}_n = \arg\min_{m_i \in \mathbf{M}'_n} \{ \sum_{m_j \in \mathbf{M}'_n} B(m_j) d(\langle m_j, m_i \rangle) \}.$$

With the above notations, we write the objective of the AMT problem as:

$$\min \max_{\mathbf{M}'_n \in \mathbf{M}'} \{ET(\mathbf{M}'_n)\}\}. \quad (12)$$

The AMT problem is also NP-hard, which can be proven by reducing a load balancing problem [33] to it by setting $d(\langle m_i, m_j \rangle)$ for all $m_i, m_j \in \mathbf{M}^{\mathbf{K}}$ to the same value. Hence, we propose a heuristic AMT algorithm as follows.

First, AMT algorithm solves the k-medoids clustering problem [34] for an initial allocation. Here, we modify the objective function of the traditional k-medoids problem into the desired observation times, i.e., $\min \sum_{n=1}^{N} \sum_{m_i \in \mathbf{M}'_n} B(m_i) d(\langle m_i, \hat{m}_n \rangle)$. We then augment the Voronoi iteration method [35] to solve the k-medoids clustering problem. That is, instead of randomly selecting initial medoids, we choose the first medoid $\hat{m}_1$ from the nodes with the highest $B(m_i)$, and iteratively select the next medoid by letting $\hat{m}_{i+1} = \arg\max_{m_i \in \mathbf{M}^{\mathbf{K}} \setminus \{\hat{m}\}} B(m_i) d(\langle m_i, \hat{m}_i \rangle)$. Second, we perform a local search to adjust the initial allocation for reducing the AMT objective value through multiple iterations. In each iteration, we generate N_e neighbors of the current allocation in one of the two ways: (i) *transfer*, in which we move one area from a subset to another, and (ii) *swap*, in which we exchange two areas originally allocated to two subsets. In either transfer or swap, we identify the best neighbor which can minimize the AMT objective function and use it as the allocation for the next iteration. We randomly select the augmentation ways in each iteration and stop whenever we exceed a user-specified maximal running time M_u, or no neighbor can improve the current allocation. The pseudo code of our AMT algorithm is given in Algorithm 1.

We note that it may not be worth to *rerun* the AMT algorithm from scratch every single time. For example, when the fire spreads, the ICS may add additional monitoring tasks. An efficient *Local Update* (LU) algorithm which greedily allocates the new tasks to the drone that leads to the minimal increase of the AMT's objective value. The LU algorithm can also be applied when changing the number of the drones. When drones are out of power, we reassign their tasks to the remaining drones also using the same LU algorithm. When additional drones are added, we run the AMT algorithm using the current allocation as the initial allocation.

The complexity of AMT algorithm is dominated by the local search method. The computational complexity for getting and evaluating a neighbor is $\mathcal{O}(|\mathbf{M}^{\mathbf{K}}|^2)$ and thus the complexity

Algorithm 1 Allocation of Monitoring Task (AMT)

Input: $\mathbf{M}^{\mathbf{K}}$, $\mathbf{K}$, number of drones N. The number of neighbors N_e in each iteration, running time limit M_u

Output: Allocation of areas $\mathbf{M}' = \{\mathbf{M}'_1, ..., \mathbf{M}'_N\}$, and task allocation: $\mathbf{K}' = \{\mathbf{K}'_1, ..., \mathbf{K}'_N\}$.

/* Step 1: Get the initial allocation. */

1 Greedily select the initial medoids $\{\hat{m}_1, ..., \hat{m}_N\}$.

2 Get the initial clustering: $\mathbf{M}' = \{\mathbf{M}'_1, ..., \mathbf{M}'_N\}$ using Voronoi Iteration [35].

3 Get $B(m_i)$ for all $m_i \in \mathbf{M}^{\mathbf{K}}$ by Eq. (10).

4 $Min \leftarrow \max_{\mathbf{M}'_i \in \mathbf{M}'} \{ET(\mathbf{M}'_i)\}$; $Sum \leftarrow \sum_{\mathbf{M}'_i \in \mathbf{M}'} \{ET(\mathbf{M}'_i)\}$.

/* Step 2: Allocation improvement. */

5 **while** $RunTime \leq M_u$ **do**

6 $num \leftarrow 0$; Shuffle list $\mathbf{M}'$ in random order.

7 **if** $Random() < 0.5$ **then**

8 **for** *pair* $(\mathbf{M}'_i, \mathbf{M}'_j) \in \mathbf{M}'$ *and* $m'_a \in \mathbf{M}'_i$ **do**

9 Get $\mathbf{M}''$ by transferring m_a to $\mathbf{M}'_j$; $num++$.

10 $Min' \leftarrow max_{\mathbf{M}'_i \in \mathbf{M}''} \{ET(\mathbf{M}'_i)\}$

11 $Sum' \leftarrow \sum_{\mathbf{M}'_i \in \mathbf{M}''} ET(\mathbf{M}'_i)$.

12 **if** $Min' < Min$ **or** ($Min' = Min$ and $Sum' < Sum$ **then**

13 $BestNeigh \leftarrow \mathbf{M}'_{\mathbf{new}}$; $Min \leftarrow Min'$; $Sum \leftarrow Sum'$.

14 **if** $num = N_e$ **then break**

15 **else**

16 **for** *pair* $(\mathbf{M}'_i, \mathbf{M}'_j) \in \mathbf{M}'$ *and* $m_a \in \mathbf{M}'_i$ *and* $m_b \in \mathbf{M}'_j$ **do**

17 Get $\mathbf{M}''$ by swapping m_a with m_b; $num++$.

18 Get $BestNeigh$ by running lines 10–14.

19 **if** $BestNeigh \neq None$ **then** $\mathbf{M}' \leftarrow BestNeigh$. **else break**

20 Get $\mathbf{K}'$ based on $\mathbf{M}'$; Return $\mathbf{M}'$ and $\mathbf{K}'$.

of the whole AMT algorithm is $\mathcal{O}(I_{max} N_e |\mathbf{M}^{\mathbf{K}}|^2)$, where N_e is the number of neighbors in each iteration, and I_{max} is the number of iterations (within M_u). Suppose there are N_a additional monitoring areas to be added, the complexity of the LU algorithm is $\mathcal{O}(N_a |\mathbf{M}^{\mathbf{K}}|^2)$.

B. Dynamic Waypoint Scheduling: DWS

Upon getting task allocation $\mathbf{K}' = \{\mathbf{K}'_1, ..., \mathbf{K}'_N\}$ from the task allocation step, the waypoint sequences of individual drones are computed in parallel for upcoming duration between t_0 and $t_0 + T$. The DWS problem generates the waypoint sequence for each drone n to maximize the weighted AUC of tasks in $\mathbf{K}'_n$. Its objective function can be written as:

$$\max \min_{k \in \mathbf{K}'_n} \{ \frac{1}{\sigma_k} \int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{Y}) dt \frac{1}{t_0 + T - g_k} \}. \quad (13)$$

We note that the DWS problem is essentially the MWSP problem with a single drone ($N = 1$). Therefore, its NP-hardness can be proved similarly. Hence, we propose a heuristic DWS algorithm which has two main steps.

In the first step, DWS algorithm adopts a classic greedy algorithm of the set cover problem [36] to select the waypoints. More precisely, we select the next waypoint that can cover the most new areas until all assigned monitoring areas are covered. Once the waypoints are chosen, we go to the Step 2, which greedily schedules the waypoints to maximize the weighted minimum AUC of all tasks from g_k to $t_0 + T$ using Eq. (6). In particular, we iteratively append the waypoint that maximizes the ratio of the improvement of the minimum waypoint AUC and the flying time. To break ties on the minimum weighted AUCs, we append the waypoint that maximizes the ratio between the number of tied tasks and the flying time. Upon appending one more waypoint, we update the AUCs

of individual tasks before getting into the next iteration. The pseudocode of our DWS algorithm is shown in Algorithm 2.

Algorithm 2 Dynamic Waypoint Scheduling (DWS)

Input: Task $\mathbf{K}'_n$, monitoring areas $\mathbf{M}'_n$, observation accuracy $\mathbf{A}_{D\times E}$, waypoint set $\mathbf{W}$, observation distance $\mathbf{d}_w$ for $w \in \mathbf{W}$, coverage matrix $\mathbf{C}$, observation distances $\mathbf{D}$.
Output: Waypoint sequence $\mathbf{P}$ for drone n within $[t_0, t_0 + T]$.

$\mathbf{P} \leftarrow [w_{in(n)}]$; $t \leftarrow t_0$, $w' \leftarrow w_{in(n)}$, $\mathbf{W}' = \emptyset$
/* Step 1: Select waypoints to cover all areas. */
for $d \in \mathbf{D}$ **do**
  $\mathbf{M}''_n \leftarrow \mathbf{M}'_n$; $\mathbf{W_d} \leftarrow \{w_i | w_i \in \mathbf{W}, d_{w_i} = d\}$.
  while $\mathbf{M}''_n \neq \emptyset$ **do**
    Get $w_i = \arg\max_{w_a \in \mathbf{W_d}} \{|\{m | m \in \mathbf{M}''_n, C(w_a, m) = 1\}|\}$.
    $\mathbf{W}'.\mathbf{add}(w_i)$; $\mathbf{M}''_n \leftarrow \mathbf{M}''_n \setminus \{m | m \in \mathbf{M}''_n, C(w_i, m) = 1\}$.
/* Step 2: Waypoint scheduling. */
while $t + d(\langle w', w_0 \rangle)/R_{fly} < t_0 + T$ **do**
  $Min \leftarrow \min_{k \in \mathbf{K}'_n} \{\frac{1}{\sigma_k} AUC(k)\}$.
  $\hat{\mathbf{M}} \leftarrow \{o_k | k \in \mathbf{K}'_n, \frac{AUC(k)}{\sigma_k} = Min\}$; $Count \leftarrow |\hat{\mathbf{M}}|$.
  /* Predict the AUC if drone visits a waypoint. */
  for $w_i \in \{w | w \in \mathbf{W}', t + \frac{d(\langle w', w \rangle) + d(\langle w, w_0 \rangle)}{R_{fly}} + T_{loi} \leq t_0 + T\}$ **do**
    Update $AUC'(k)$ with $k \in \{k' | k' \in \mathbf{K}'_n, C(w_i, o_{k'}) = 1\}$ if drone n visit w_i next by Eq. (6).
    $Min[i] \leftarrow \min_{k \in \mathbf{K}'_n} \{\frac{1}{\sigma_k} AUC'(k)\}$.
    $\hat{\mathbf{M}}[i] \leftarrow \{o_k | k \in \mathbf{K}'n, AUC'(k)/\sigma_k = Min[i]\}$.
    $Count[i] \leftarrow |\hat{\mathbf{M}}[i]|$.
  /* Select the next waypoint. */
  if $\min_{w_i \in \mathbf{W}} \{Min[i]\} < Min$ **then**
  $\rho \leftarrow \arg\max_{w_i \in \mathbf{W}'} \frac{Min - Min[i]}{d(\langle w', w_i \rangle)/R_{fly} + T_{loi}}$.
  else $\rho \leftarrow \arg\max_{w_i \in \mathbf{W}'} \frac{Count - Count[i]}{d(\langle w', w_i \rangle)/R_{fly} + T_{loi}}$.
  /* Update t, w', $\mathbf{P}$ and the AUC of tasks. */
  $t \leftarrow t + d(\langle w', \rho \rangle)/R_{fly} + T_{loi}$; $w' \leftarrow \rho$; $\mathbf{P}.\mathbf{add}(\rho)$.
  for $k \in \{k' | k' \in \mathbf{K}'_n, C(\rho, o_{k'}) = 1\}$ **do**
    Update $l_k(t)$, $R_k(l_k(t))$ and $AUC(k)$ by Eq. (6).
return $\mathbf{P}$

Besides, we propose a DWS variant algorithm for faster coverage. The idea is to visit the monitoring tasks with 0 AUCs first before considering other tasks. More concretely, each drone flies to the waypoint that maximizes the ratio between the covered tasks and the flying time. Once no monitoring task has 0 AUC, we run DWS algorithm to complete the waypoint sequence. We refer to this algorithm as DWSF.

The computational complexity of the waypoint selection step of DWS is $\mathcal{O}(N_c |\mathbf{W}|^2)$, where N_c is the maximum number of areas a waypoint can cover. The complexity of the waypoint scheduling step of DWS is $\mathcal{O}(T |\mathbf{W}'| |\mathbf{K}|)$, where $\mathbf{W}'$ is the number of selected waypoints. The time complexity of the DWS and DWSF algorithms is $\mathcal{O}(N_c |\mathbf{W}|^2 + T |\mathbf{W}'| |\mathbf{K}|)$.

VI. Evaluations

In this section, we evaluate the performance of our proposed algorithms for solving the Allocation of Monitoring Tasks (AMT) and Dynamic Waypoint Scheduling (DWS) problems. We refer to these two problem as *allocation* and *scheduling* problems in our discussion for brevity.

A. Simulator Implementations and Setup

We have implemented a detailed simulator in Python, which is modularized and can work with different allocation and scheduling algorithms. Because Multi-agent Traveling Salesman Problem (MTSP) is a special case of our MWSP, we choose representative near-real-time MTSP techniques as our baseline algorithms for comparison. More specifically, we have implemented the K-Medoids (KM) algorithm [34] to compare with our AMT algorithm for solving the task allocation problem. The KM algorithm clusters monitoring areas using Euclidean distance. For the waypoint scheduling problem, in addition to our proposed DWS and DWSF algorithms, we also have implemented: (i) the Minimum Improvement (MI) algorithm, which greedily selects the waypoint that improves the task with the lowest AUC in each iteration, (ii) the Nearest Neighboring (NN) algorithm, which generates a recurring TSP tour using a nearest-neighboring approximation [37], and (iii) the Minimum Spanning Tree (MST) algorithm, which also generates a recurring TSP tour using a minimum spanning tree approximation [38]. We consider all pairs of the allocation and scheduling algorithms, as illustrated in Table IV.

TABLE IV: Considered Algorithmic Combinations

Allocation \ Scheduling	DWS	DWSF	MI	NN	MST
AMT	AMT-DWS	AMT-DWSF	AMT-MI	AMT-NN	AMT-MST
KM	KM-DWS	KM-DWSF	KM-MI	KM-NN	KM-MST

We estimate the observation accuracy levels of several concerned events using the experiment results in the literature [25], [26]. Table V gives the estimated observation accuracy of five representative events. For realistic simulations, we consider a building with 12 floors and 384 windows, as illustrated in Fig. 4. For each simulation, several rooms are randomly chosen as the fire sources. Each room has a 10% probability to have humans, and each window has a 5% probability to be open. We simulate the fire dynamics using the fire spread model with recommended parameters [39]. Moreover, humans may be trapped in a room that is close to a fire scene. Otherwise, humans randomly leave rooms with random states. If not otherwise specified, we generate random states using normal distributions.

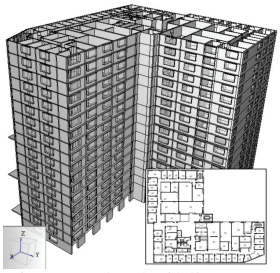


Fig. 4: The building used in our simulations.

TABLE V: Observation Accuracy

Event (e_k)	Acc. $A(15, e_k)$	Acc. $A(5, e_k)$
Fire	0.70	0.99
Fire Intensity	0.38	0.89
Human	0.69	0.98
Human Activity	0.37	0.90
Open Window	0.32	0.80

According to the practical monitoring requirements [10], we split the whole drones surveillance process into two phases: discovery and monitoring phases. During the discovery phase, the ICS strives to get an overview of the high-rise fire scene by detecting fires, humans, and open windows in the building. Next, we enter the monitoring phase, except for the above tasks, ICS also monitors the changes of the fire intensity

TABLE VI: Task Types

Type	Event (e_k)	Sig. (σ_k)	Fre. (η_k)
1	Fire	1	0.2
2	Fire	3	0.5
3	Fire Intensity	2	0.5
4	Human	1	0.25
5	Human	3	0.5
6	Human Movement	2	1
7	Open Window	1	0.2
8	Open Window	3	0.5

TABLE VII: Simulation Parameters

Para.	Value	Para.	Value
R_{fly}	$3\ m/s$	T_{loi}	$5\ s$
$\mathbf{D}$	$5\ m, 15\ m$	N_e	100
f_5^H	$3.26\ m$	T_s	$30\ s$
f_5^W	$2.18\ m$	f_{15}^W	$9.79\ m$
f_{15}^H	$6.54\ m$	M_u	$40\ s$

and tracks the human movements. We set following rules to generate monitoring tasks, based on the task types in Table VI.

(a) During discovery phase, the types 1, 4, 7 tasks are added at all windows in the building.
(b) If a fire source is reported, types 2, 5, and 8 tasks are added for all windows on that floor and the floor above.
(c) If fire is reported in a room, type 3 tasks are added at the room's windows; if humans are detected in a place, types 6 tasks are added for covering windows there.
(d) If human is no longer observed in a room, types 4, 5 or 6 tasks in that room are removed; if open window is detected, types 7 and 8 tasks are removed.

We run each simulation for 30 minutes, with each plan duration is $T = 5$ minutes. During the simulation, we record all events in the perception table and check its state every 1 minute to generate new tasks for the new perceptions. The recorded events include the presence of fires, humans, open windows in the discovery phase. In addition, the changes of fire intensity and human movements are also recorded in the monitoring phase. For the overall performance per simulation run, we adopt a sampling rate of T_s. The performance metrics in our simulation are:

- *Missing events:* The number of undetected events based on the perception table at each time instance.
- *Minimum weighted AUC:* MWSP's objective function.
- *Weighted accuracy:* We compute the minimum accuracy among the tasks in each significance level. We then compute the weighted accuracy across all significance levels.
- *Weighted reliability:* We consider a task is reliable if its accuracy exceeds a threshold Θ_d. We then calculate the ratio of reliable tasks in each significance level. We define weighted reliability as weighted ratio across all significance levels.
- *Running time:* Computation time of the algorithms.

Table VII summarizes the key parameters adopted in our simulations. We also vary several parameters in our simulations, including the number of tasks between 96 and 314, and the number of drones between 3 and 12 to study the scalability of our solution. Our simulation parameters are empirically chosen for the high-rise fire situation at hand. For example, we stop increasing the number at 12 drones since the resulting missing events are very few. For statistically meaningful results, we repeat each experiment 25 times and report the average results with 95% confidence intervals if applicable.

B. Simulation Results

Scheduling algorithms. Fig. 5 compares our proposed DWSF with other baseline algorithms when using the AMT algorithm for task allocation. We give sample results from simulations with 5 drones and 177 tasks. Figs 5(a)–(d) reveal that our AMT-DWSF algorithm always outperforms the baseline scheduling algorithms in all aspects throughput a sample simulation run. Figs 5(e)–(h) report the overall performance from diverse scheduling algorithms across 25 runs. These figures depict that, on average, the AMT-DWSF algorithm achieves 33% fewer missing events, 39 times gain on weighted minimum AUC, 1.2 times gain on weighted accuracy, and 2.8 times gain on the weighted reliability, compared with the AMT-MST algorithm. Fig. 5 reveals that scheduling algorithms solely based on Euclidean distance (NN and MST) cannot handle heterogeneous tasks at fire scenes. It is more evident in the monitoring phase (after about 10 minutes) when monitoring tasks have higher heterogeneity. *In contrast, the DWSF algorithm better accommodates the task heterogeneity and exercises the trade-off between the coverage and accuracy for better performance.* Another interesting observation on Fig. 5(a) is that the slope of accumulative missing events is steeper when the simulation just starts. This is because the perception table is empty initially; once fires, humans, and open windows are detected after the drones visited all monitoring areas at least once, the missing events only occur when fire scene states change. Fig. 5(b) also shows that the minimum weighted AUC is 0 during the first 8 minutes, which indicates that not all tasks have been performed until the 8-th minute. Overall, our proposed AMT algorithm performs well in both discovery and monitoring phases.

Allocation algorithms. Fig. 6 compares the performance of the two allocation algorithms (AMT and KM) with two representative scheduling algorithms (DWSF and DWF). We report sample results from simulations with 5 drones and 177 tasks. We omit the other three scheduling algorithms since they give similar results. Figs. 6(a)–(d) present the average performance results across the 25 simulations. Compared to KM, our AMT algorithm always delivers fewer missing events, higher minimum weighted AUC (20% improvement on average), higher weighted accuracy (16% increase on average), and higher weighted reliability (46% boost on average) when working with DWS and DWSF algorithms. *Fig. 6 reveals the superior performance of our proposed AMT algorithms, compared to the baseline ones.*

Scalability of our proposed algorithm. We next consider larger MWSP problems with different number of drones (between 3 and 9) and different number of tasks (between 96 to 314). First, we employ the AMT algorithm for the allocation problem and compare the performance of scheduling algorithms in Fig. 7. Figs. 7(a) and (b) present the performance of different scheduling algorithms under different number of drones. Both the figures reveal that as the number of drones increases, our AMT-DWSF algorithm has fewer accumulative missing events (by up to 42%), and higher minimum weighted

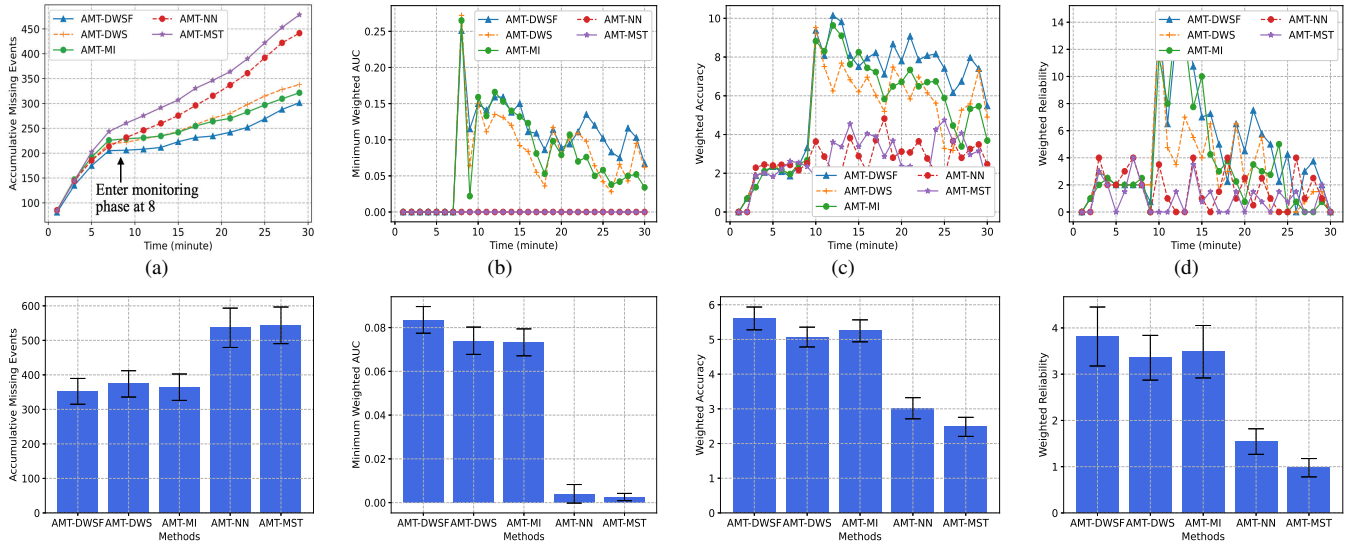


Fig. 5: Performance of DragonFly (integrated AMT-DWSF algorithm) on: (a), (e) accumulative missing events, (b), (f) minimum weighted AUC, (c), (g) weighted accuracy, and (d), (h) weighted reliability.

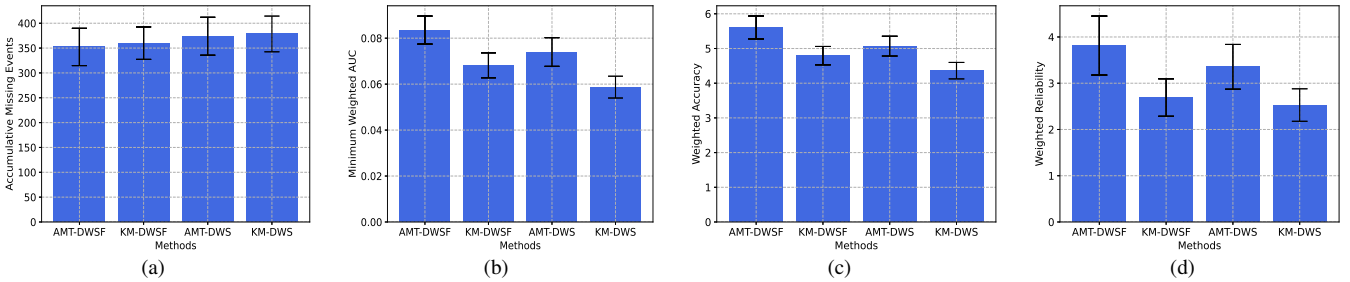
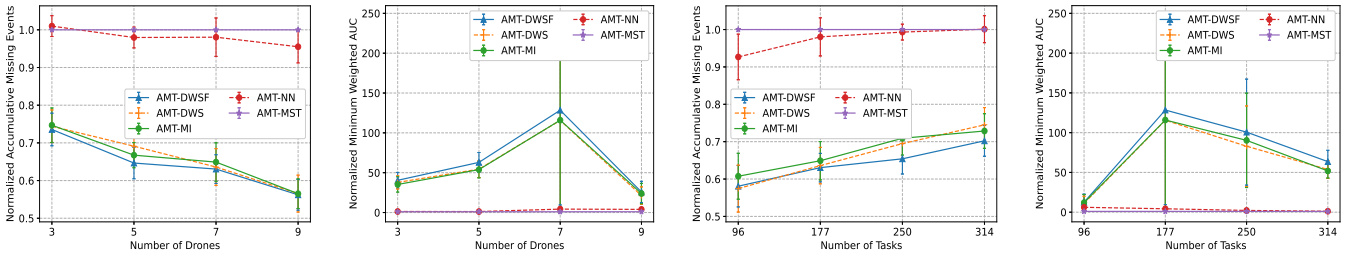


Fig. 6: Impact of task allocation strategy on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability.

AUC (by up to 110 times), compared to the AMT-MST algorithm. Figs. 7(c) and (d) depict the statistics of the various scheduling algorithms under different number of tasks. Our AMT-DWSF algorithm always results in the lowest number of accumulative missing events (by up to 40%), and the highest minimum weighted AUC (by up to 110 times) compared with those of the AMT-MST algorithm. *From Fig. 7, we validate that our proposed AMT-DWSF outperforms other scheduling algorithms under diverse number of drones and tasks.*

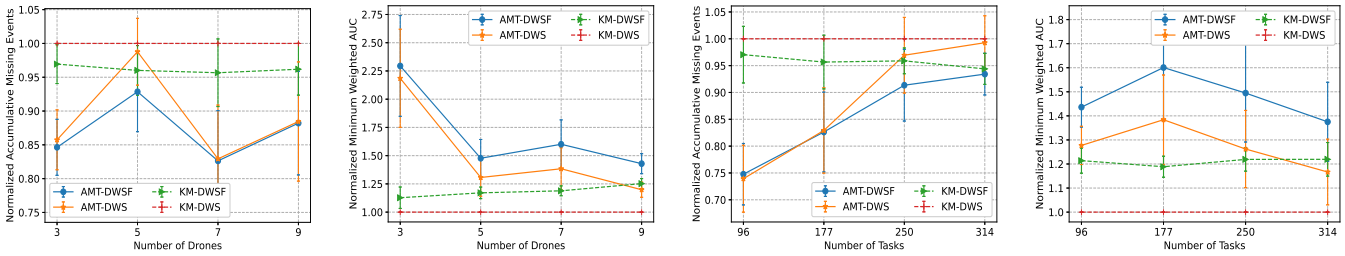
Next, we explore the performance of the AMT algorithm in different scenarios and report the results in Fig. 8. Figs. 8(a) and 8(b) show the normalized value of the accumulative missing events and the minimum weighted AUC by that of KM-DWS algorithms. These figures show that as the number of drones increases, the AMT algorithm always leads to fewer missing events (by up to 15%), higher minimum weighted AUC (by up to 87%), compared to those of the KM algorithm using the same scheduling algorithm. It also delivers higher weighted accuracy (by up to 20%), higher weighted reliability (by up to 92%), compared with the KM-DWS algorithm (figures not shown for brevity). Figs. 8(c) and 8(d) compare the AMT and KM algorithms with 7 drones under the different number of tasks normalized to the KM-DWS algorithm. These figures show that the AMT algorithm constantly outperforms the KM algorithm when the number of tasks increases. For example, with 177 tasks, the accumulative missing events achieved by AMT-DWSF is only about 87% of the KM-DWSF algorithm. AMT-DWSF also results in higher weighted minimum AUC (about 1.3 times) than theKM-DWSF, achieves higher weighted accuracy (about 1.2 times), and higher weighted reliability (about 1.9 times) than KM-DWSF (some figures are not shown due to space limit). *Fig. 8 demonstrates that our proposed AMT algorithm offers superior performance under different (larger) problem size.* We also observe that the performance gain of the AMT algorithm shrinks as the number of tasks increases. This is partially due to the limited running time for the local search.

Last, we investigate the running time of our proposed algorithms. Here, we focus on the LU algorithm, because the full-fledged AMT is only invoked once in each simulation, which can be done offline in about 40 seconds. Fig. 9 reports the running time of the LU algorithm plus the waypoint scheduling algorithms under diverse scenarios. The running times are collected from an Intel i7 workstation. This figure



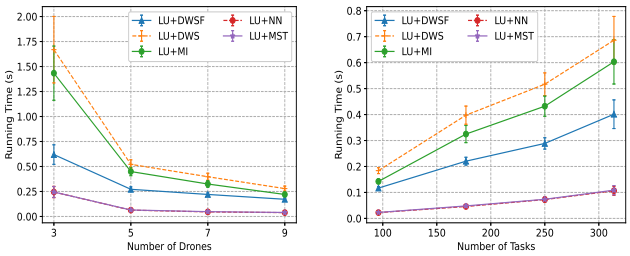
(a) (b) (c) (d)

Fig. 7: Performance of DragonFly (integrated AMT-DWSF algorithm) under: (a), (b) 177 tasks with 3 to 9 drones, (c), (d) 7 drones with 96 to 314 tasks. (a), (c) give normalized accumulated missing events, and (b), (d) give normalized minimum weighted AUC.



(a) (b) (c) (d)

Fig. 8: Impact of task allocation strategy under diverse problem sizes: (a), (b) 177 tasks with 3 to 9 drones, (c), (d) 7 drones with 96 to 324 tasks.(a), (c) give normalized accumulated missing events, and (b), (c) give normalized minimum weighted AUC.



(a) (b)

Fig. 9: Total running time of the LU and scheduling algorithms under different numbers of (a) drones and (b) tasks. Sample results from simulations with (a) 177 tasks and (b) 7 drones.

clearly shows that our LU and DWSF algorithms terminate within 0.6 second, which is negligible compared to the plan duration that lasts for minutes. Moreover, from Fig. 9(b), we also observe that the running time of our LU and DWSF algorithms increases linearly to the number of tasks. *Hence, Fig. 9 depicts that our LU and DWSF algorithms react fast even with larger problem size.*

VII. Conclusion and Future Work

In this paper, we designed Dragonfly, a framework for guiding drones to monitor dynamic fire scenes. We formulated and developed techniques for solving the Multi-drone Waypoint Scheduling Problem (MWSP) to have multiple drones continuously monitor high-rise fires with both coarse- and fine-grained observations for optimal overall accuracy. The MWSP problem is NP-hard, and we proposed a two-step solution that first allocated tasks/regions among drones, and then scheduled the assigned waypoints of each drone. Our extensive evaluations demonstrated the superior performance of our solution under heterogeneous tasks at a dynamic fire scene. Future work will involve the integration of vision processing techniques into the workflow to better tune the complexity associated with event detection to where it is most needed. Novel image processing workflows and models are also required to capture fire severity using AI-enabled methods to localize building features (e.g., windows or doors) on the fly. The use of multi-sensor fusion approaches can help improve coverage-accuracy tradeoffs. We also plan to leverage formal methods to reason about the tradeoffs between the reliability and timeliness of data collection, especially in mission-critical scenarios [40], [41]. Considering aerial sensing in the areas of poor network conditions, such as wildfire monitoring, we plan to establish IoT platforms for drones, based on Software Defined Networking (SDN) technologies [42]–[44] to enable effective data collection and transmission in the scenarios of heterogeneous sensing requirements and access technologies, such as LTE, 5G, WiFi, and ZigBee.

Acknowledgment

The authors thank Amanda Kimball at Fire Protection Research Foundation, Bart van Leeuwen at Netage B.V., Prof. Marco Levorato and his team, members in DSM, TIPPERS, and Sparx-Cal projects at UCI, for their valuable input on the work. We acknowledge Katelyn Itano and Sarika Rau for initial building models on this effort.

REFERENCES

[1] FireRescue1, "5 Drone Technologies for Firefighting," Mar. 2014, https://tinyurl.com/y4ldl8mf, Last accessed on 2019-11-10.

[2] Measure a 32 Advisor Company, "Drones for Disaster Response and Relief Operations," Apr. 2015, https://tinyurl.com/yy2pp6tc,.

[3] Sarah Calams, "6 takeaways on how fire departments are using drones and the barriers preventing purchase," 2018, https://tinyurl.com/y2qhs43x,.

[4] R. Nakata and C. et al., "Rf techniques for motion compensation of an unmanned aerial vehicle for remote radar life sensing," in *Intl. Microwave Symp.*, 2016.

[5] National Institute for Occupational Safety and Health (NIOSH),, "Three Fire Fighters Die in a 10-Story High-Rise Apartment Building - New York," August 1999, https://tinyurl.com/y63p84j7,.

[6] M. et al., *Fire fighting tactics under wind driven conditions: laboratory experiments*. Fire Protection Research Foundation, 2009.

[7] N. White and M. Delichatsios, *Fire hazards of exterior wall assemblies containing combustible components*. Springer, 2015.

[8] M. Moore Bick, "Grenfell tower inquiry: Phase 1 report overview - report of the public inquiry into the fire at grenfell tower on 14 june 2017," 2019, https://www.grenfelltowerinquiry.org.uk/phase-1-report.

[9] National Institute of Justice, "A Guide for Investigating Fire and Arson," May 2009, https://tinyurl.com/yy48k9wq.

[10] United States Fire Administration National Fire Academy, "Incident Command for Highrise Operations ICHO-Student Manual," 2006.

[11] C. Reardon and J. Fink, "Air-ground robot team surveillance of complex 3D environments," *Intl. Symp. on Safety, Security and Rescue Robotics*, pp. 320–327, 2016.

[12] E. et al., "UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development," *Intl. Conf. on Unmanned Aircraft Systems*, pp. 274–283, 2014.

[13] A. Ilah N. Alshbatat, "Fire Extinguishing System for High-Rise Buildings and Rugged Mountainous Terrains Utilizing Quadrotor Unmanned Aerial Vehicle," *Intl. Journal of Image, Graphics and Signal Processing*, vol. 10, no. 1, pp. 23–29, 2018.

[14] P. Pecho, P. Magdolenová, and M. Bugaj, "Unmanned aerial vehicle technology in the process of early fire localization of buildings," *Transportation Research Procedia*, vol. 40, pp. 461–468, 2019.

[15] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," *Intl. Conf. on Intelligent Agent Technology*, 2004.

[16] A. Wallar, E. Plaku, and D. A. Sofge, "Reactive Motion Planning for Unmanned Aerial Surveillance of Risk-Sensitive Areas," *IEEE Transactions on Automation Science and Engr.*, vol. 12, 2015.

[17] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," *IEEE Conf. on Decision and Control*, 2010.

[18] V. Sea, A. Sugiyama, and T. Sugawara, "Frequency-based multi-agent patrolling model and its area partitioning solution method for balanced workload," *Lecture Notes in Computer Science*, vol. 10848 LNCS, 2018.

[19] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of unmanned aerial vehicles*. Springer, 2015.

[20] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative-timing missions," *Journal of Guidance, Control, and Dynamics*, vol. 28, 2005.

[21] R. W. Beard and T. W. M. et al., "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, 2002.

[22] Y. Kim, D. W. Gu, and I. Postlethwaite, "Real-time optimal mission scheduling and flight path selection," *IEEE Transactions on Automatic Control*, vol. 52, no. 6, pp. 1119–1123, 2007.

[23] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem," in *Intl. Conf. on Decision and Control*, vol. 2, 2003.

[24] S. Leary, M. Deittert, and J. Bookless, "Constrained UAV mission planning: A comparison of approaches," *Proceedings of the IEEE Intl. Conf. on Computer Vision*, pp. 2002–2009, 2011.

[25] N. Lakshminarayana, Y. Liu, K. Dantu, V. Govindaraju, and N. Napp, "Active face frontalization using commodity unmanned aerial vehicles," *arXiv preprint arXiv:2102.08542*, June 2021.

[26] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals using stereo imagery for accurate object class detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1259–1272, May 2017.

[27] J. Shermeyer and A. Van Etten, "The effects of super-resolution on object detection performance in satellite imagery," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[28] Netage B.V., "Smart Data for Smarter Firefighters," https://netage.nl/resc-info/.

[29] G. Yang and X. e. a. Lin, "A telecom perspective on the internet of drones: From lte-advanced to 5g," *arXiv preprint*, 2018.

[30] DJI, "Mobile SDK," https://developer.dji.com/mobile-sdk/.

[31] M. N. Bygi, "3D Visibility Graph," *Computer Engr.*, 2007.

[32] M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.

[33] J. Kleinberg and E. Tardos, *Algorithm Design*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[34] E. Schubert and P. J. Rousseeuw, "Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms," in *Similarity Search and Applications*, G. e. a. Amato, Ed. Springer Intl. Publishing, 2019.

[35] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, 2009.

[36] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem," *Operations research*, vol. 47, 1999.

[37] P. M. e. a. França, "The m-traveling salesman problem with minmax objective," *Transportation Science*, vol. 29, no. 3, pp. 267–275, 1995.

[38] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part ii," *Mathematical programming*, vol. 1, 1971.

[39] H. Cheng and G. V. Hadjisophocleous, "Dynamic modeling of fire spread in building," *Fire Safety Journal*, vol. 46, pp. 211–224, 2011.

[40] N. Venkatasubramanian, C. Talcott, and G. A. Agha, "A formal model for reasoning about adaptive qos-enabled middleware," *ACM Trans. Softw. Eng. Methodol.*, vol. 13, no. 1, p. 86â€“147, Jan. 2004.

[41] G. Denker, N. Dutt, S. Mehrotra, M.-O. Stehr, C. Talcott, and N. Venkatasubramanian, "Resilient dependable cyber-physical systems: a middleware perspective," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 41–49, 2012.

[42] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[43] K. E. Benson, G. Bouloukakis, C. Grant, V. Issarny, S. Mehrotra, I. Moscholios, and N. Venkatasubramanian, "Firedex: A prioritized iot data exchange middleware for emergency response," in *Middleware Conference*, 2018, pp. 279–292.

[44] K. E. Benson, G. Wang, N. Venkatasubramanian, and Y.-J. Kim, "Ride: A resilient iot data exchange middleware leveraging sdn and edge cloud resources," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018, pp. 72–83.