

UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Leveraging Mobility to Enhance IoT Applications

Permalink

<https://escholarship.org/uc/item/3w97t90v>

Author

Liu, Fangqi

Publication Date

2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,  
IRVINE

Leveraging Mobility to Enhance IoT Applications

DISSERTATION

submitted in partial satisfaction of the requirements  
for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Fangqi Liu

Dissertation Committee:  
Professor Nalini Venkatasubramanian, Chair  
Professor Marco Levorato  
Professor Sang-Woo Jun

2023



# TABLE OF CONTENTS

	Page
<b>LIST OF FIGURES</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>vii</b>
<b>ACKNOWLEDGMENTS</b>	<b>viii</b>
<b>VITA</b>	<b>x</b>
<b>ABSTRACT OF THE DISSERTATION</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 IoT Applications and Techniques: An Overview . . . . .	1
1.2 Challenges and Concerns in IoT Deployment . . . . .	5
1.3 Using Mobility to Enhance IoT Applications . . . . .	8
1.4 Thesis Contributions and Organization . . . . .	9
<b>2 Related Work</b>	<b>13</b>
2.1 Characteristics of IoT Applications . . . . .	13
2.2 Mobility of Humans and Animals . . . . .	15
2.2.1 Enhancing IoT Sensing with Human and Animal Mobility . . . . .	16
2.2.2 Enhanced Networking with Human or Animal Mobility . . . . .	18
2.3 Mobility of Ground Vehicles . . . . .	19
2.3.1 Mobility-Enhanced IoT Sensing with Ground Vehicles . . . . .	19
2.3.2 Ground Vehicles and IoT Networking . . . . .	20
2.4 Mobility of Aerial Vehicles (UAV/Drones) . . . . .	21
2.4.1 Mobility-Enhanced IoT Sensing with UAV and Drones . . . . .	22
2.4.2 Aerial Vehicles in Data Transmission . . . . .	23
2.5 Guiding Mobile Entities: Insights from Various Research Fields . . . . .	24
<b>3 Approach Overview</b>	<b>27</b>
3.1 Time-Sensitive Community IoT Applications . . . . .	27
3.2 Opportunistic vs. Planned Mobility . . . . .	29
3.3 Leveraging Planned Mobility to Enhance IoT . . . . .	31
3.3.1 Scenario 1: Mobility for Cost-Effective Network Coverage . . . . .	33
3.3.2 Scenario 2: Utilizing Mobility to Enhance Sensing Coverage . . . . .	34

3.3.3	Scenario 3: Mobility for Enhanced Networking and Sensing . . . . .	36
<b>4</b>	<b>Cost-Effective Data Transmission with Public Transportation Fleets</b>	<b>39</b>
4.1	Chapter Overview . . . . .	40
4.2	Sample Scenario and Problem Statements . . . . .	42
4.3	Problem Formulation . . . . .	45
4.3.1	Symbols and Notations . . . . .	45
4.3.2	Formulation . . . . .	49
4.4	Solution Approach and Algorithms . . . . .	50
4.4.1	Upload Point Placement Algorithms . . . . .	51
4.4.2	Upload Path Planning Algorithms . . . . .	54
4.5	Experimental Evaluation of our Approach . . . . .	56
4.5.1	Scenarios . . . . .	57
4.5.2	Comparison Results . . . . .	58
4.6	Summary and Discussion . . . . .	64
<b>5</b>	<b>DragonFly: Drone-Assisted High-Rise Monitoring for Fire Safety</b>	<b>66</b>
5.1	Chapter Overview . . . . .	67
5.2	Tackling the High-Rise Fire Scene . . . . .	70
5.3	The DragonFly Framework . . . . .	72
5.4	Multi-Drone Coordination for High-Rise Fires . . . . .	75
5.4.1	Monitoring Tasks . . . . .	75
5.4.2	Candidate Waypoints . . . . .	76
5.4.3	Accuracy of Monitoring Tasks . . . . .	77
5.4.4	Formulation of the Multi-Drone Waypoint Scheduling Problem (MWSP)	79
5.5	Proposed Algorithms for MWSP . . . . .	83
5.5.1	Allocation of Monitoring Tasks: AMT . . . . .	84
5.5.2	Dynamic Waypoint Scheduling: DWS . . . . .	88
5.6	Evaluations . . . . .	90
5.6.1	Simulator Implementations and Setup . . . . .	90
5.6.2	Simulation Results . . . . .	93
5.7	DragonFly Implementation . . . . .	101
5.7.1	System Architecture . . . . .	102
5.7.2	Prototype Implementation and Experiments . . . . .	103
5.8	Summary and Discussion . . . . .	106
<b>6</b>	<b>DOME: Drone-assisted Monitoring of Emergent Events For Wildland Fire Resilience</b>	<b>109</b>
6.1	Chapter Overview . . . . .	110
6.2	Problem Definition and Approach . . . . .	112
6.3	Physics-inspired Task Generator . . . . .	118
6.4	Multi-drone Flight Planning . . . . .	123
6.4.1	Symbols and Notations . . . . .	123
6.4.2	Spatial-temporal Factors for Task Execution . . . . .	127
6.4.3	Formulating MFP . . . . .	128

6.5	Proposed Algorithms for MFP . . . . .	129
6.5.1	Step 1: Allocating Tasks to Drones . . . . .	129
6.5.2	Step 2: Single Drone Flight Planning . . . . .	133
6.6	Experimental Evaluation . . . . .	138
6.6.1	Simulation setup . . . . .	138
6.6.2	Experimental Results . . . . .	141
6.7	System Implementation . . . . .	146
6.8	Summary and Discussion . . . . .	150
<b>7</b>	<b>Conclusion</b>	<b>151</b>
7.1	Summary of Thesis Contributions . . . . .	151
7.2	Key Observations and Insights . . . . .	154
7.3	Future Work . . . . .	158
	<b>Bibliography</b>	<b>163</b>

# LIST OF FIGURES

	Page
3.1 Categories of Mobility . . . . .	31
3.2 IoT System Architecture with Selected Components . . . . .	32
3.3 Challenges and Solutions for IoT Applications in Three Explored Scenarios . . . . .	38
4.1 Sensor data collection with scheduled fleets. . . . .	42
4.2 Our approach with two collaborating algorithms. . . . .	50
4.3 Orange country bus routes and stops . . . . .	55
4.4 Comparisons the performance between the FC and DM algorithms with GA and UPS algorithms under different cost limitations (a) data delivery ratio, (b) late delivery ratio, and (c) data transfer time. . . . .	59
4.5 Performance of the four upload point placement algorithms under different cost limitations: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio. . . . .	61
4.6 Installation cost and running time of the four upload point placement algorithms under different cost limitations: (a) total cost, (b) number of UPs and (c) running time. . . . .	62
4.7 The performance gains of UPS comparing with the other algorithms on: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio under different cost limitations in the scenario with 40 RPs. . . . .	63
5.1 Overview of DragonFly framework. . . . .	73
5.2 Sample $IA_k(t)$ with $\eta_k = 0.2$ and $g_k = 0$ . . . . .	79
5.3 Workflow of our proposed algorithms. . . . .	84
5.4 The building used in our simulations. . . . .	91
5.5 Performance of DragonFly throughout a sample simulation on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability. . . . .	94
5.6 Performance of DragonFly across 25 runs on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability. . . . .	95
5.7 Impact of task allocation strategy on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability. . . . .	97
5.8 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. . . . .	98

5.9	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. . . . .	100
5.10	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. . . . .	101
5.11	System Structure of DragonFly System. . . . .	102
5.12	Mock-up Building Facade with Fires . . . . .	104
5.13	Dashboard of DragonFly System . . . . .	105
5.14	Real implementation with one fire source: (a) Expected and real fly time (b) Real trajectory with different algorithm . . . . .	106
5.15	Performance of our DWSF Algorithm in Real Implementation on (a) Weighted Accuracy, (b) Weighted Reliability, and (c) Minimum Weighted AUC. . . . .	107
6.1	Our Previous Rx Burns. . . . .	113
6.2	Overview of DOME Framework. . . . .	114
6.3	Workflow of Task Generator. . . . .	119
6.4	Task Generation Rules . . . . .	121
6.5	Rules for event-driven task update . . . . .	122
6.6	Illustration of three burn sites and fire ignition strategy. . . . .	139
6.7	Subtask number in burn sites ① (a),② (b) under diverse wind speeds . . . . .	142
6.8	Performance of our UTA-DFP algorithm at Burn sites ②, (a) gives total reward and (b) gives total missing subtasks. . . . .	142
6.9	Performance of our UTA-DFP at Burn sites ③. (a) gives the total reward, and (b) the total missing subtasks. . . . .	143
6.10	Performance of (a), (b) task allocation and (c), (d) flight planning algorithms. (a), (c) give the total reward, (b), (d) give the missing subtasks. . . . .	144
6.11	Performance of (a), (b) UTA and (c), (d) FDP algorithms. (a), (c) give the total reward, (b), (d) give the missing subtasks. . . . .	145
6.12	Running time of UTA-DFP at burn sites ① (a) and ② (b) . . . . .	146
6.13	System Architecture of DOME . . . . .	147
6.14	Sensors, Testbed and Dashboard in DOME . . . . .	148
6.15	Drone-based Mobile Sensing in DOME System . . . . .	148



# LIST OF TABLES

	Page
4.1 Considered Algorithms . . . . .	57
4.2 Simulation Parameters . . . . .	58
5.1 Initial Perception . . . . .	74
5.2 Perception at time 10:05 . . . . .	74
5.3 Task Table . . . . .	74
5.4 Considered Algorithmic Combinations . . . . .	91
5.5 Observation Accuracy . . . . .	91
5.6 Task Types . . . . .	92
5.7 Simulation Parameters . . . . .	92
6.1 Data quality score under diverse PPMs . . . . .	125
6.2 Simulation Parameters . . . . .	140

# ACKNOWLEDGMENTS

As I stand on the threshold of completing my Ph.D. journey, I am deeply moved to express my appreciation to those remarkable individuals who have been pivotal in shaping my academic path.

First and foremost, I am deeply thankful to my advisor, Professor Nalini Venkatasubramanian. Her unwavering support, guidance, and encouragement have been the cornerstone of my Ph.D. experience. Professor Venkatasubramanian's mentorship has not only shaped my research but has also helped me persist in the pursuit of knowledge. I am truly fortunate to have had the opportunity to learn from her.

I am indebted to Professor Cheng-Hsin Hsu at National Tsing Hua University (NTHU). His guidance and friendship have been invaluable in shaping my research path and enriching my understanding of the field.

To my coauthors on the referred publications and the peers with whom I closely collaborated during the early stage of my Ph.D. journey – Qiuxi Zhu, Professor Md Yusuf Sarwar Uddin, Qing Han, Guoxi Wang, Hang Nguyen, Nailah Saleh Alhassoun, and Praveen Venkateswaran – your collective efforts and shared knowledge have profoundly enriched my research experience. Your friendship and companionship were indispensable during my days at UCI and have been integral to my personal growth. I also would like to thank my current labmates – Tung-Chun Chang, Andrew Chio, Rahul Atul Bhope, Ryan Hildebrant, Rummana Rahman, Modeste Mefenya Kenne, and Yuqiao Li. Our friendship and the lasting memories we've created are truly remarkable to me.

I extend my gratitude to all the professors I have had the privilege of knowing at UCI. Professor Sharad Mehrotra, thank you for your continuous guidance and valuable suggestions. I also appreciate the enjoyable moments of hosting parties with Nalini. Professor Marco Levorato, thank you for your guidance in the realm of wireless communication and drone experiments. Professor Tirtha Banerjee, I am thankful for your guidance in conducting experiments in the forest, which was a unique experience. I extend my appreciation to my committee members, Professor Amelia Regan and Professor Sang-Woo Jun, for your valuable suggestions and guidance throughout my Ph.D. journey. During my internship at SRI, I am grateful to my mentors Minyoung Kim and Carolyn Talcott for their kindness and guidance. I also want to thank everyone in the Information Systems Group (ISG) at the University of California, Irvine, for their significant contributions and valuable feedback.

To my friends I've met since coming to UCI – Yiming Lin, Zhongyue Luan, Xinwen Zhang, Yuan Tao, Qiushi Bai, Koti Allu, Janine Ann Baijnath-Rodino, Shu Li, Joy Fan, and all those with whom I collaborated with and met in my previous life – I deeply appreciate your presence in my life and the enriching experiences we've shared.

To my family, your continuous support has been the bedrock of my journey to this point.

My Ph.D. research is supported in part by the National Institute of Standards and Tech-

nology (NIST) under award No. 70NANB17H285, the University of California Office of the President (UCOP) under award No. LFR-20-653572, the United States Air Force and DARPA under awards No. FA8750-16-2-0021 and No. FA8750-16-C-0011, the United States Navy and DARPA under awards No. N66001-15-C-4065, No. N66001-15-C-4067 and No. N66001-15-C-4070, the National Science Foundation (NSF) under award No. 2008993, and the Donald Bren School of Information and Computer Sciences (ICS) at the University of California, Irvine (UCI).

# VITA

Fangqi Liu

## EDUCATION

<b>Doctor of Philosophy in Networked Systems</b> University of California, Irvine	<b>2023</b> <i>Irvine, California</i>
<b>Master of Software Engineering</b> Jilin University	<b>2017</b> <i>Jilin, China</i>
<b>Bachelor of Software Engineering</b> Jilin University	<b>2014</b> <i>Jilin, China</i>

## RESEARCH EXPERIENCE

<b>Graduate Research Assistant</b> University of California, Irvine	<b>2017–2023</b> <i>Irvine, California</i>
--	---

## TEACHING EXPERIENCE

<b>Teaching Assistant</b> University of California, Irvine	<b>2019–2022</b> <i>Irvine, California</i>
<b>Graduate Reader</b> University of California, Irvine	<b>2017–2019</b> <i>Irvine, California</i>

## WORK EXPERIENCE

<b>Research Intern</b> SRI International, Computer Science Lab	<b>Summer 2021</b> <i>Menlo Park, California</i>
---	---

## AWARDS

<b>CSP Rising Star</b> CPS Rising Stars 2023 Workshop	<b>May 2023</b> <i>Charlottesville, VA</i>
--	---

## REFEREED CONFERENCE PUBLICATIONS

- Cost-Effective Sensor Data Collection from Internet-of-Things Zones Using Existing Transportation Fleets** **June 2019**  
IEEE International Conference on Smart Computing (SMARTCOMP)
- DragonFly: Drone-Assisted High-Rise Monitoring for Fire Safety** **July 2021**  
International Symposium on Reliable Distributed Systems (SRDS)
- WinSet: The First Multi-Modal Window Dataset for Heterogeneous Window States** **Nov. 2021**  
ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys) (Short Paper)
- Enhancing Situational Awareness with Adaptive Fire-fighting Drones: Leveraging Diverse Media Types and Classifiers** **Oct. 2022**  
ACM Multimedia Systems Conference (MMSys)
- DOME: Drone-assisted Monitoring of Emergent Events for Wildland Fire Resilience** **May 2023**  
14th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)
- Demo Abstract: DOME – IoT-Based Monitoring Emergent Events for Wildland Fire Resilience** **May 2023**  
ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI) (Demo)

## REFEREED JOURNAL PUBLICATIONS

- Distance-Driven Consensus Quantification** **August 2017**  
IEEE Transactions on Intelligent Transportation Systems
- ADMB: Application-driven multi-hop broadcast for vehicular networks** **Dec. 2017**  
International Journal of Communication System

# ABSTRACT OF THE DISSERTATION

Leveraging Mobility to Enhance IoT Applications

By

Fangqi Liu

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2023

Professor Nalini Venkatasubramanian, Chair

The Internet of Things (IoT) has revolutionized the world, finding widespread utilization across diverse fields, industries, healthcare, and city management. Despite its success, the exponential growth of the IoT market has presented challenges for traditional IoT systems that demand attention. In urban scenarios, the significant influx of data from media-rich sensors poses a strain on local networks with limited resources, impacting their efficiency and effectiveness. Additionally, the impracticality and high cost of deploying sensors and network infrastructures in remote areas creates obstacles to IoT deployment. This thesis aims to explore the utilization of mobile entities as a solution to address these challenges. Specifically, we emphasize the usage of planned entities with predictable or controllable movements to enhance the sensing and networking capabilities of IoT systems, particularly in time-sensitive community IoT applications. Time-sensitive scenarios demand data collection and analysis within specific time frames to preserve data value. To comprehensively investigate the usage of mobility, this thesis explores strategies for integrating mobile entities in diverse scenarios across urban and remote areas, each presenting unique time-sensitivity requirements and design challenges.

In the first scenario, we investigate the utilization of public transit fleets in network-constrained smart city applications. Our proposal involves using these fleets and network infrastructures

along their routes to establish a cost-effective backbone network for long-range sensor data transmission, effectively addressing the limitations of local network resources. To achieve this, we develop approaches for optimal deployment of network infrastructure along with planning data collection from public transit fleets, considering the heterogeneity of delay tolerance and priority of sensor data, as well as the trade-off between data delivery delay/loss and network infrastructure installation cost. This thesis evaluates the proposed approaches using real-world bus networks in Orange County, CA and compares them with several other methods.

As a second use case, we investigate the use of drones to enhance sensing coverage in mission-critical IoT applications, specifically high-rise fire monitoring, where in-situ sensors are unavailable due to extreme conditions. We design and implement a drone-based IoT platform for real-time data collection in fire settings. The platform provides a dashboard for firefighters to visualize monitoring areas, user interfaces for commanding tasks to drones, and automatic flight planning for multiple drones to fulfill specified monitoring tasks. We propose multiple-drone flight planning approaches, optimizing data collection processes while considering the heterogeneity of monitoring tasks in terms of periods and priorities, as well as the trade-off between sensing coverage and data quality. The proposed algorithms are evaluated in a simulated high-rise fire scenario with real building structures at UCI. Additionally, we assess the applicability of the proposed system by implementing it in a lab-based testbed with mockup high-rise fires.

In the third scenario, we focus on utilizing drones to assist mobile sensing and sensor data transmission of IoT-based monitoring systems in remote areas with limited in-situ sensors and poor network conditions, particularly for wildland fire monitoring. We automate the drone-based monitoring system by enabling real-time perception of the physical world based on sensor data, automatic task generation for mobile entities, and dynamic planning and control of their movements to continuously monitor dynamic environments. We propose

a rule-based task generation procedure for spatial-temporal monitoring requirements based on fire status and prediction. Additionally, we investigate approaches for multiple-drone flight planning, considering data collection timeliness, the trade-off between sensing coverage and data quality, and network disconnection during flights. The proposed flight planning algorithms are evaluated using simulated wildland fire burns at the Blodgett Forest Research Station, and the system's applicability is assessed through lab-based testbed implementation.

Overall, this thesis offers valuable insights into using mobile entities to address challenges in traditional IoT systems and enhance time-sensitive IoT applications. The exploration of different scenarios, from leveraging public transit fleets as a cost-effective backbone network to employing drones for high-rise fire monitoring and remote area sensing, demonstrates the versatility and practicality of mobile solutions in advancing IoT technologies.



# Chapter 1

## Introduction

In today's interconnected world, the Internet of Things (IoT) has emerged as a technological phenomenon that has captured significant attention from researchers and businesses, leading to a wave of innovation across industries. The rapid advancements in IoT have transformed how we interact with our environment and have opened up new possibilities by redefining the boundaries of connectivity and profoundly reshaping our daily lives. In this section, we provide an overview of community-based IoT applications, which are the focus of this thesis. We discuss their primary techniques and examine the challenges they encounter. Building upon this foundation, the main contributions of this thesis involve exploring the role of mobility in enhancing IoT applications, with a particular focus on data transmission and mobile sensing aspects. Through the lens of three driving use cases, we delve into the potential of leveraging mobility to optimize IoT functionality and address key challenges in IoT deployment and implementations.

### 1.1 IoT Applications and Techniques: An Overview

The Internet of Things (IoT) can be described as a network of physical devices (e.g., vehicle, street lights, and buildings) embedded with sensors, software, and other technologies that enable data collection and exchange over a variety of communication channels [130]. Since its initial introduction in 1999, Internet of Things (IoT) [109] has transformed the process of data collection by enabling automated data gathering through interconnected devices. Today, IoT applications have experienced widespread adoption across various sectors such as industry, healthcare, and community services, establishing their popularity and ubiquity [178]. This thesis centers on community-based IoT applications [186], which refer to interconnected smart technologies designed for specific neighborhood and community needs. These applications address local challenges and offer customized services to residents, applicable across diverse sectors from urban to remote areas.

In urban scenarios, IoT-driven smart cities [236] enhance the quality of life through efficient monitoring of environmental factors like air quality, noise level, traffic, waste, and energy consumption, which are facilitated by automated control systems. The influence of IoT extends to smart homes [49], where connected devices create intelligent and energy-efficient living spaces. Home automation has grown to allow remote control of lighting, heating, security systems, and entertainment. In the healthcare domain [76], IoT enables connected medical devices, wearable sensors, and remote monitoring systems, which revolutionize patient care with continuous health tracking, personalized plans, and preventive care. In rural areas, IoT enables applications such as smart agriculture [144] which helps farmers optimize irrigation, fertilizer application, and harvests for increased productivity. Additionally, IoT supports wildlife and vegetation monitoring [95], which contributes to conservation efforts. The integration of IoT in smart grids has modernized energy management and helped foster sustainability and development in remote communities. In disaster response [56], the IoT has enabled surveillance capabilities to aid emergency responders in making informed decisions and deploying resources efficiently.

The rapid development of IoT over the past 20 years can be attributed to several key techniques, which have also served as the primary inspiration for this thesis. Next, we will outline the fundamental techniques driving the evolution of IoT.

- **Miniaturization of hardware.** The proliferation of the IoT has been greatly facilitated by the development and advancement of compact and energy-efficient hardware platforms such as Arduino [86] and Raspberry Pi [215] since 2013. The trend towards smaller and cheaper compute components, sensors, and microcontrollers have contributed to the feasibility of embedding these technologies into diverse objects and devices. This miniaturization has resulted in the widespread adoption of IoT devices across various domains, from small wearable devices to industrial sensors.
- **Improvement of connectivity.** Advancements in communication technologies have also been instrumental in shaping the evolution of the IoT. The earlier stages of IoT relied on Radio Frequency Identification (RFID) technology for applications like automatic identification of goods and location tracking [96]. As the number of devices grew and the IoT became increasingly pervasive in society, concerns about exhausting the IPv4 address space were raised. The adoption of IPv6 by major Internet service providers in 2012 marked a significant milestone and ensured the availability of a substantial IP address space to enable the connection of numerous IoT devices, making the future of IoT a reality [190]. Following this, the advancement of wireless networks, low-power protocols (e.g., blacktooth, Zigbee), and cellular connectivity (e.g., 4G, 5G) have provided reliable and scalable connectivity options for seamless data exchange between devices and systems [128]. Furthermore, the exploration of vehicular ad-hoc networks and delay-tolerant networks [9] has presented new opportunities for extending the range of IoT communication to support flexible and adaptable connectivity among mobile devices. This expands the reach of IoT to new applications in diverse environments.

- **Cloud computing and big data analysis.** Cloud computing has become a natural solution to handle the growing volume of IoT data by providing a scalable and efficient way for processing, storage and analysis [84]. By integrating cloud computing with IoT, organizations can offload compute and storage requirements from IoT devices to the cloud, which helps relieve the burden on the devices themselves. Additionally, cloud platforms offer scalability, allowing businesses to expand their IoT deployments effectively. Simultaneously, big data analysis has also emerged as a powerful tool to process the ever-increasing volume, velocity, and variety of data generated by IoT devices. Various techniques support big data analysis, including distributed processing frameworks like Apache Hadoop [15] and Apache Spark [235]. These frameworks rely on parallel computing, distributed data storage, and load balancing across clusters, thereby accelerating data processing procedures. Furthermore, a multitude of matured data mining, stream processing, and machine-learning-based algorithms work to extract insights from IoT data.
- **Availability of unmanned vehicles.** Unmanned vehicles, specifically UAVs and drones, have emerged as a pivotal component of the IoT ecosystem by facilitating data acquisition and extending the reach of IoT applications to new heights. Since the mid-2010s, consumer-grade drones started gaining popularity due to improved drone technology, miniaturized sensors, and improvement of connectivity [224]. With the rise of companies like DJI and Parrot, consumer-oriented drone manufacturers began producing affordable, ready-to-fly drones with basic camera capabilities. Subsequently, the utilization of drones in IoT gained significant attention. Nowadays, the utilization of drones has broadened into areas such as infrastructure inspection, disaster response, environmental monitoring, and public safety. Drones equipped with advanced sensors, cameras, and connectivity capabilities provide real-time data and insights for better decision-making, remote monitoring, and improved operational efficiency.

- **Integrating control into IoT.** In Industrial Internet of Things (IIoT) applications, precise control is crucial to facilitate the automated management of diverse processes and systems. This has been achieved through the integration of Cyber-Physical Systems (CPS) control loops into IoT systems [151]. CPS employs closed-loop control, bridging physical systems with computational and communication technologies [130]. This integration unlocks new automation, optimization, and intelligent decision-making possibilities for organizations. IoT establishes the foundation for seamless data collection and real-time monitoring, enabling CPS to proactively track device conditions and ensure efficient control. This combination also fosters autonomous and adaptive systems, where IoT sensors supply real-time data, empowering CPS to autonomously respond, adapt, and optimize physical processes using predefined rules or machine learning algorithms. This autonomy results in improved efficiency, reduced human intervention, and dynamic responses to changing conditions.

## 1.2 Challenges and Concerns in IoT Deployment

The increasing demand for community-based IoT applications, along with the advancements in IoT techniques, has given rise to both new opportunities and significant challenges. Next, we provide an outline of current challenges, with a focus on concerns related to the design and implementation of IoT systems. These challenges will serve as the primary focus of our research work, and we will address them in detail throughout this thesis.

**(C1) Big data volume vs. network resource limitation.** The availability of sufficient network resources for long-range data transmission is a significant challenge in implementing IoT in urban scenarios. The rapid increase in the number of IoT devices equipped with media-rich sensors (e.g., microphones, cameras, Radar/Lidar sensors, etc.) generates massive volumes of sensor data. By 2025, the total number of IoT devices will reach 22 billion [24],

which will generate an exponential amount of data, based on growing trends of 0.1 *zettabytes* in 2013 and 79.4 *zettabytes* by 2025 [209, 163]. This large amount of data obtained from IoT devices needs to be fused, analyzed, and interpreted, which often requires high-bandwidth networks to transmit the data from devices to big data processing backends for comprehensive analysis. However, such solutions can be costly and challenging to implement since access networks usually have limited bandwidth, leading to network congestion. Additionally, while cellular networks and emerging technologies like 5G can potentially complement existing network infrastructure, they may not be cost-effective. Network deployments, especially in remote or rural areas, can be expensive and may pose challenges in achieving comprehensive IoT connectivity.

**(C2) Availability of IoT devices in challenged areas.** Another challenge in IoT deployments is the lack of network and sensor availability, particularly in remote or hard-to-reach areas. Deploying and maintaining IoT devices in such locations can be difficult and costly, often due to the absence of necessary network infrastructure, such as the unavailability of cellular networks in forested areas. The rapid deployment of IoT devices in emergency surveillance scenarios immediately following a disaster also presents significant challenges. Furthermore, in extreme conditions like wildfires, deploying sensors becomes impractical due to the inaccessibility of certain locations for humans and the potential risk of sensor damage. Together, these issues make it challenging to gather real-time information about situations and impede the implementation of IoT solutions.

**(C3) Scalability and adaptability.** Given the pervasive nature of IoT applications and the rapidly evolving landscape, IoT devices and systems may need to be deployed, relocated, or adapted to operate in different locations under changing circumstances, depending on a specific application's requirements. For example, in a smart city environment, the deployment of IoT sensors and devices may need to be adjusted to address changing traffic patterns, urban development, or environmental factors. Moreover, IoT applications for

monitoring dynamic environments like wildland fires require careful deployment of IoT devices to ensure coverage of the constantly changing landscape. Here, the challenge lies in ensuring the scalability of IoT systems to accommodate the growing number of devices and data, as well as the adaptability to quickly respond to updates in the monitoring area and diverse monitoring objectives. Overcoming these challenges enables efficient and effective monitoring, empowering timely decision-making and resource allocation.

**(C4) Time sensitivity.** One key challenge in IoT applications is the timing of data collection and transmission, typically involving various levels of delay tolerance that vary across different applications. In some IoT applications, real-time data collection and transmission are critical. Mission-critical scenarios, such as emergency response systems or real-time monitoring of industrial processes, demand immediate and continuous data updates to ensure timely decision-making and rapid response. Ensuring the availability of real-time data in such applications requires careful consideration of deployment strategies and network infrastructure design. Conversely, there are scenarios where certain data types have a higher tolerance for delays. For instance, in applications involving the long-term analysis of phenomena like air quality or building energy consumption, data is periodically collected and can withstand delays of days or even up to a week. Optimizing the deployment of IoT devices and network resources to meet the specific timing requirements of different data types in a cost-effective manner becomes a significant concern.

**(C5) Action planning and coordination of mobile devices.** The introduction of unmanned vehicles such as drones and rovers has enhanced the capabilities of the IoT but has also brought forth challenges regarding the effective and efficient plan and control of mobile devices to fulfill specific data collection tasks. When scheduling the trajectory of these devices, multiple factors must be considered, including the timing and method of data collection, the location of data capture, and the unique characteristics of the devices themselves (i.e., movement speed, activity range, sensor capabilities, etc.). Coordinating the movements

of multiple mobile devices and allocating specific tasks to them requires efficient planning and resource management. The challenge lies in developing strategies and algorithms to assign tasks, optimize their movement patterns, and ensure effective collaboration among these mobile devices.

This thesis acknowledges several unaddressed challenges in IoT deployment. Specifically, it does not cover security and privacy concerns in IoT deployment, nor does it address the aspects of energy consumption and maintenance of IoT devices.

### **1.3 Using Mobility to Enhance IoT Applications**

This thesis aims to investigate the role of mobility in community-based IoT applications to address the aforementioned challenges. By leveraging the advantages of mobility, such as pervasive coverage, increased flexibility, and fast movement, the goal is to enhance the efficiency, effectiveness, and overall performance of IoT systems in diverse applications.

Firstly, incorporating mobile devices can reduce IoT infrastructure deployment costs by leveraging their extensive coverage range. In urban scenarios, for example, mobile objects such as vehicles and crowds can create delay-tolerant or vehicular ad hoc networks. This eliminates the need for in-situ network infrastructures for long-range and massive data transmission. Additionally, the effective sensing range can be significantly extended by employing sensors carried by humans or mounted on vehicles, eliminating the necessity for widespread sensor deployment.

Secondly, mobile devices greatly enhance the scalability and adaptability of IoT, particularly in challenging areas where network connectivity and sensing devices are limited. In disaster scenarios, aerial vehicles carrying sensors can provide a top-down perspective of monitoring areas that are inaccessible to ground-based sensors. This flexibility in movement offered by



mobile devices allows for comprehensive and efficient monitoring.

Furthermore, mobile devices are easy to deploy and can provide timely responses in emergency scenarios where the deployment of sensors and network infrastructures is infeasible. For example, in fire settings, firefighters can promptly fly drones to the fire source to track the fire status and detect trapped people. The agility and responsiveness of mobile devices enable quick action and situational awareness in critical situations.

## 1.4 Thesis Contributions and Organization

Considering the advantages of exploring mobile devices in IoT systems, this thesis aims to explore the usage of mobility in IoT applications to enhance sensing and networking capabilities. To explore the various functionalities of mobility, we have designed approaches to plan the actions of mobile entities in three main scenarios, each with unique application contexts and challenges to be addressed. Firstly, we leverage mobility to assist in long-range sensor data transmission in urban settings. Secondly, we investigate the usage of flexible aerial mobile devices to extend sensor coverage during emergency monitoring in urban scenarios. Lastly, we persist in utilizing mobile aerial devices to expand sensing capabilities and facilitate real-time data transmission in applications focused on monitoring emergent events in remote areas where the network conditions are challenging. In the following, we will illustrate the organization of this thesis and highlight its main contributions in each chapter.

- Chapter 2 surveys the related work about the usage of mobility in IoT applications.
- Chapter 3 presents an overview of our approach, providing details on the main challenges in the three explored scenarios and the strategies for leveraging mobility to enhance IoT capabilities accordingly.

- Chapter 4 details our contribution in exploring the usage of the mobility of public transit fleets, sensors, and network infrastructures to create a cost-effective backbone network in urban scenarios facilitating sensor data collection and transmission in IoT applications. To support this design, we propose algorithms for network infrastructure deployment and data collection planning to optimize data transmission. Here, we consider the time sensitivity of different types of sensor data and the cost of network infrastructure installation. Our approaches are evaluated using a real-world bus network in Orange County, CA, and the efficiency of the proposed method is compared to several other heuristic approaches. The results demonstrate the superior performance of our algorithm. For instance, in a specific scenario, our approach operates with a worst case of  $\sim 21$  seconds (a  $15\times$  improvement) in data transfer time, a 3.2% of packages delivered beyond the delay tolerance ( $12\times$  reduction), and 96% data delivery ratio ( $\sim 50\%$  improvement).
- Chapter 5 delves into our design and implementation of a drone-based IoT platform, which utilizes drone mobility to support real-time sensor data collection in mission-critical IoT applications in urban scenarios, using high-rise fire as a driving use case. We design an end-to-end platform, DragonFly, to facilitate the visualization of situational events based on data analysis results, provide user interfaces for humans to determine monitoring needs, and support drone-based data collection for continuous event monitoring. To support this system, we introduce multiple-drone flight planning strategies aimed at guiding drones in collecting sensor data to address diverse monitoring objectives. This encompasses the gathering of information to enhance sensor coverage (event identification) and data accuracy (obtain fine-grained data for improved event detection). We plan the flights of multiple drones through two primary steps: 1) allocating monitoring tasks to individual drones, and 2) dynamically scheduling waypoints to determine optimal drone routes. To evaluate the proposed approaches, we conduct a performance evaluation employing a simulated high-rise fire scenario that incorporates

a realistic fire spread model. We then compare the performance of our proposed algorithms with baseline techniques. Simulation results emphasize the superiority of the proposed algorithms. In particular, DragonFly algorithms achieve a 33% reduction in the number of undetected events and a  $39\times$  enhancement in data accuracy compared to the baseline algorithms. Moreover, we implemented the proposed system and conducted experiments using real drones in a lab-based testbed to evaluate the system’s applicability and the performance of the proposed algorithms.

- Chapter 6 explores the usage of drones to improve the sensing and network capabilities of IoT systems, focusing on wildland fire monitoring as a driving use case. We introduce the DOME system, tailored for drone-based monitoring endeavors. DOME encompasses a data analysis component for real-world perception, an automated task generation procedure, and a multi-drone flight planning module to manage task completion. To support the DOME system, we design a rule-based task generation procedure based on formal logic to establish spatial-temporal monitoring requirements for drones, leveraging real-time fire status perception and physics-based fire spread models. Furthermore, we investigate multiple drone flight planning strategies, integrating algorithms for task allocation (mapping tasks to drones) and individual drone flight path planning. These strategies aim to schedule the drone sensing process to meet various time-sensitive demands for monitoring dynamic features while also ensuring a balance between sensing coverage and data quality. Simultaneously, these approaches enable drones to store and upload data, effectively addressing network disconnectivity during flights. We evaluate the DOME system using simulated prescribed fires based on planned burns at Blodgett Forest Research Station. We then compare our proposed algorithms with various baseline methods. Our experiments show the effectiveness of DOME’s integrated mechanisms with respect to data quality (measured by total reward) and task completion (reduction in missing subtasks) when compared to baseline algorithms. Specifically, our proposed algorithm achieves a  $1.7\times$  increase in total re-

ward and a 99% reduction in missing subtasks compared to the baseline algorithms. Furthermore, we implement the proposed system in a lab-based testbed with mockup wildland fires to rigorously evaluate its applicability and effectiveness.

- Chapter 7 concludes this thesis by presenting the valuable lessons we have learned throughout our research journey. We offer a holistic view of our proposed approaches to support mobility-enhanced IoT systems and suggest future work to further enhance the utilization of mobility in IoT applications.

# Chapter 2

## Related Work

This chapter seeks to gather valuable insights into the utilization of mobility to enhance Internet of Things (IoT) applications across various entities. We will explore the mobility of humans, animals, ground vehicles, unmanned robots, and aerial vehicles (UAVs and drones). Specifically, we will discuss how these diverse entities' mobility supports a wide range of IoT applications from networking and sensing perspectives and discuss the opportunities and challenges of integrating them into IoT applications.

### 2.1 Characteristics of IoT Applications

In this section, we explore community-based IoT applications [186] that involve the deployment of IoT technologies within localized neighborhoods, aiming to improve community safety, sustainability, and quality of life of its residents. These applications utilize IoT devices for data collection and analysis, enabling informed decisions through data-driven solutions to optimize community services. In order to better abstract the characteristics of diverse IoT applications, we categorize them based on two key factors: spatial locations and data

collection time sensitivity.

### **Urban vs. Remote Areas**

In urban scenarios characterized by high population density and extensive development, IoT devices and network infrastructures are readily available and pervasive. Additionally, the integration of mobile entities, such as crowds, vehicles, and transportation fleets, further augments data collection and transmission capabilities in this context. This setting offers numerous opportunities for diverse IoT applications. Smart cities [236, 206] use IoT to monitor conditions, optimize traffic, manage waste, track energy consumption [79, 208, 197], and control street lighting. Smart homes [49, 140, 204] enhance residents' quality of life through automated control of appliances, lighting, heating, and security systems. Environment monitoring IoT applications [217, 146, 139] collect data on air quality, noise levels, temperature, and pollution, aiding urban planning and sustainability efforts. Additionally, IoT systems in emergency response [12, 171] enable real-time alerts and swift responses for incidents like fires, ensuring safety for residents and efficient emergency services.

Conversely, in remote areas, IoT applications are characterized by limited network infrastructure and in-situ IoT devices. Examples of these applications include smart farming and agriculture [144, 98], where IoT is used to optimize crop management and resource utilization in remote farming areas. Wildlife monitoring and forest monitoring [95, 187, 136] are also essential applications, leveraging IoT devices to track and protect wildlife and assess ecological conditions in distant and challenging terrains. Additionally, IoT plays a crucial role in disaster scenarios, such as wildfire and flood monitoring [56, 242, 78], where real-time data collection aids in rapid alerts and effective disaster management in remote regions.

### **Mission-Critical vs. Delay-Tolerant Scenarios**

Mission-critical applications are those that require real-time data collection, analysis, and response due to their time-critical nature. Examples of these applications include emergency

response scenarios, such as monitoring and responding to fire incidents, conducting human rescues, and managing disaster situations [135, 78]. In the healthcare sector, mission-critical IoT applications like remote patient monitoring and medical alert systems rely on real-time data to detect and respond promptly to critical health events or emergencies [76, 182], providing timely medical assistance and intervention. Additionally, transportation management systems [237, 20] utilize real-time data collection and analysis to monitor traffic flow, detect accidents, and coordinate emergency response for road incidents, contributing to safer and more efficient traffic control.

Delay-tolerant applications, on the other hand, indicate applications where data collection and analysis are relatively tolerant of delays. Examples of such applications include environmental monitoring, where IoT deployments collect data on air quality, weather patterns, and pollution levels [37, 73, 196]. Additionally, IoT systems for public transportation tracking can be designed to tolerate delays, providing updates on bus or train schedules that account for delays lasting minutes to hours. This enables commuters to plan their journeys effectively [43, 17]. In the field of weather forecasting, IoT-enabled weather stations may also operate on delay-tolerant systems, offering data updates at regular intervals.

In the following sections, we will discuss how diverse mobile entities serve the above applications based on their unique characteristics.

## **2.2 Mobility of Humans and Animals**

We first examine the mobility of humans and animals in the context of IoT applications, investigating how it enhances IoT capabilities in sensing and network perspectives.

### 2.2.1 Enhancing IoT Sensing with Human and Animal Mobility

In urban scenarios, the mobility of humans, when coupled with mobile devices equipped with various sensors, actively contributes to sensor data collection over a wider activity range. This has given rise to a powerful approach known as Crowdsensing [39, 232], which finds extensive use in monitoring various factors for diverse applications. One exemplary platform is Atmos [160], which leverages a Crowdsensing network of mobile devices to collect weather-related sensor data. By combining this data with human input, Atmos assists in human localization and weather prediction. Another urban monitoring system [37] takes advantage of mobile phones' cellular network connectivity status and the position of buses and taxis to provide real-time evaluation of urban dynamics to understand the movement of people and vehicles. Project NoiseTube [138] is another notable example that utilizes mobile phones as noise sensors. The system encourages users to share their geo-localized noise measurements and provide additional personal annotations. Similarly, authors in [175] have designed and implemented Ear-Phone, an urban noise mapping system. This system leverages crowdsourcing to collect noise samples from mobile users and utilizes compressive sensing techniques to reconstruct the noise map based on the gathered data.

Moreover, humans themselves can serve as "sensors" by providing valuable social media data, which can be harnessed for multiple purposes across various IoT applications. For instance, Crooks et al. [53] conducted an analysis of spatial-temporal characteristics of Twitter feed activity in response to an earthquake in the United States to identify and localize the impact area of the event. Here, they regard Twitter as an effective distributed sensor system for event detection and impact assessment. Similarly, Takeshi et al. [185] leveraged utilized real-time interactions on Twitter to detect events, such as earthquakes, and proposed an approach to localize the event's center. Longueville et al. [59] explored the usage of Twitter data to support emergency planning and risk/damage assessment during a major forest fire event in the South of France. Bengtsson et al. [27] introduced an innovative approach that



leverages position data of mobile phone SIM cards to estimate the magnitude and trends of population movements following the Haiti 2010 earthquake and cholera outbreak. Furthermore, Ballesteros et al. [22] utilized spatial-temporal indexed crime datasets to predict future crime and safety index values of specific locations based on past crime events. They further incorporated mobile devices and geo-social networks to record user trajectory traces, enabling personalized, context-aware safety recommendations.

The mobility of animals presents intriguing opportunities for IoT applications in remote areas. By attaching specialized sensor devices to animals, data can be collected from otherwise inaccessible areas for multiple applications such as wildlife conservation and ecological research. This approach, known as “animal-borne sensing,” [174] empowers researchers and conservationists to gain valuable insights into environmental conditions and animal behaviors by collecting data directly from the animals themselves. In this context, there are two notable platforms for animal monitoring that collect sensor data by leveraging animals to carry sensors for monitoring diverse features. The first platform is the “e-Pasto Platform” [2], which offers an application for monitoring cattle grazing on mountains. The second platform is “Digitanimal” [1], which provides a range of GPS-based devices for tracking various animal species, including cattle, goats, sheep, and dogs. Moreover, Arshad et al. [16] introduced a smart dairy-monitoring system, equipping farm cows with temperature sensors, stethoscopes for heart rate tracking, and GPS modules to monitor environmental conditions and animal status. The system also enables automatic control of farm functions, such as a water-filling unit for drinking water. Similarly, Chaudhry et al. [42] presented an IoT-based real-time system for livestock health monitoring. The system involves mounting custom-designed multi-sensor boards on livestock to record physiological parameters and incorporates a camera for behavioral pattern identification. The system effectively monitors parameters such as skin temperature, heart rate, and rumination. Furthermore, Mitra et al. [145]) has undertaken an effort in wildlife monitoring by equipping animals with advanced sensors. The deployed sensors include cameras to capture images of animals and GPS sensors to track

their precise locations. Subsequently, drones are deployed to collect data from these sensors using wireless network techniques. Antonio Ferreira Cardoso et al. conducted the SheepIT project [41], which aims to detect unwanted behaviors and enable sheep to graze in vineyards without human supervision. To achieve this, each sheep is equipped with an electronic collar containing sensors that retrieve posture, activity, and localization data. This technology ensures the safety of both the vines and grapes. Additionally, these collar sensors also assist in monitoring the sheep's health status and identifying abnormal situations, such as potential predator attacks.

### **2.2.2 Enhanced Networking with Human or Animal Mobility**

The mobility of humans and animals has also been utilized to address intermittent connectivity challenges in IoT applications by creating Delay-Tolerant Networks (DTNs) or assisting data delivery. Recent research [28] has highlighted the important role of human mobility in augmenting the capabilities of DTNs, enabling seamless data transmission and exchange between IoT devices. The examples include [82] proposed architecture that leverages opportunistic communication among humans to enhance information forwarding and dissemination within IoT communities. In [225], the authors designed interfaces that enable opportunistic communications and interactions among mobile users and smart objects, aiming to enhance connectivity in IoT systems. The utilization of animal mobility has also been explored to create DTN for assisting data transmission in IoT. For example, in the ZebraNet project [95], wild zebras were equipped with special GPS collars capable of continuously relaying their data to encountered neighbors. The data was forwarded until reaching the base station, where it was utilized for wildlife monitoring purposes. Ayele et al. [18] proposed an opportunistic dual radio network architecture that utilizes animals mounted with sensors for wildlife tracking and data collection. In this approach, mobile animals serve as data relayers, facilitating the transmission of data from sensors to remote gateways via LoRa technology.

## 2.3 Mobility of Ground Vehicles

In this section, we will explore how the mobility of ground vehicles in urban scenarios, including cars, bicycles, and public transit such as buses, trains, and railways, has been utilized in IoT applications for efficient sensor data collection and transmissions.

### 2.3.1 Mobility-Enhanced IoT Sensing with Ground Vehicles

Vehicles are commonly utilized as sensor carriers in IoT to collect data in urban scenarios due to their wide activity range. Their ability to traverse various locations within dynamic and densely populated urban environments enables them to capture diverse and valuable data points, making them an ideal choice for monitoring and data collection. Numerous related works leverage these entities in IoT applications. For example, in [43], GPS data from taxis is used to track human mobility in cities, facilitating the planning of night-bus routes. Additionally, buses are employed as sensor carriers in the Mosaic system [73], aiming to monitor PM2.5 in urban environments. Similarly, Kang et al. [101] developed an IoT platform named M-ESB, utilizing buses to carry sensors for monitoring urban environments, including parameters such as air quality and road conditions. Caminha et al. [55] proposed SensingBus, a system employing buses to carry sensors for data collection in urban scenarios, primarily for environmental monitoring purposes. Buses transport the collected data to fog nodes, supporting long-range data transmission to the cloud. Moreover, Jorge Lanza et al. [124] explored strategies for utilizing public transportation fleets, taxis, and vehicles to support mobile sensing in urban scenarios. They investigated the network deployment required to facilitate this distributed data-gathering process and developed a testbed in the city of Santander to validate their approach.

Unmanned Ground Vehicles (UGVs) have emerged as valuable tools for enhancing mobile sensing capabilities in IoT applications, providing a diverse array of innovative solutions and applications across various domains. For instance, Marek et al. [169] explored the usage of UGVs equipped with a number of sensors, including Velodyne VLP-16 LiDAR, a stereo camera, an IMU, and a GPS, to capture data for 3D mapping in forests. Similarly, Tomasz et al. [34] implemented an IoT system using a mobile robotic vehicle equipped with a sensing system consisting of a LIDAR and an IMU to achieve 3D mapping of isolated Industrial Terrain. Tao et al. [159] utilized UGVs to achieve accurate localization and real-time road information perception using sensors such as global navigation satellite systems, LiDAR, and IMU. Their work proposes a method for road profile estimation using LiDAR and vehicles with an active suspension system, as well as a control method for vehicle navigation on the roads. Zhu et al. [247] introduced autonomy into agricultural vehicles by equipping them with multiple sensors, such as LIDAR, IMU, and encoders for attitude prediction. Building on this setup, their work explores future attitude prediction and designs a control approach to guide the motion of UGVs in order to prevent rollover and maintain safe operations on unstructured terrains. Moreover, the usage of UGVs with diverse sensors in agricultural environments also includes detection of animal fecal matter, surveys of crop growth, detection of crop damage, and identification of pests or molds [31]. Navigation of these UGVs is typically accomplished using vision-based cameras and GPS units. This combination of sensors allows for precise data collection and analysis, enabling farmers to make informed decisions regarding crop management and pest control, leading to increased efficiency and improved yields.

### **2.3.2 Ground Vehicles and IoT Networking**

The mobility of ground vehicles has also been extensively employed to complement traditional network infrastructures in IoT systems, leading to improved reliability, flexibility, and network coverage. For instance, Zarafshanarak et al. [238] proposed TrainNet, a vehicular

network that utilizes trains to transport nonreal-time data. In this network, railway stations and trains are equipped with storage devices, enabling the efficient transfer of bandwidth-intensive digital data, such as music, video, and movies, among nearby communities. Buses have also been employed to enhance data transmission in the KioskNet [83] project, where they are dispatched to collect data from edge servers (PCs) deployed in kiosks in rural scenarios. The collected data is then transported to remote gateways, enabling cost-effective communication. Zhu et al. have introduced the SCALECycle [248] IoT platform, which employs bicycles to support mobile sensing and data collection in community-based IoT applications with intermittent and varying sensing/communication coverage. The BikeNet project [63] utilizes sensors installed on a cyclist's bicycle to collect quantitative ride data, with data collection occurring through opportunistic wireless access points, facilitating delay-tolerant transmission. Additionally, real-time communication and transmission of sensor data rely on the cyclist's mobile phone's cellular data channel. Another example is CarTel [90], a mobile sensor computing system designed to collect, process, deliver, and visualize data from sensors located on mobile units such as automobiles. Furthermore, Zguira et al. [239] have introduced the "Internet of Bikes" IoB-DTN protocol which applies the Delay/Disruption Tolerant Network paradigm to support Internet of Things (IoT) applications running on urban bike sharing system-based sensor network. We have provided additional detailed discussion regarding research efforts involving the use of ground vehicles and buses for data collection and transmission assistance in Section 4.2.

## 2.4 Mobility of Aerial Vehicles (UAV/Drones)

In addition to ground vehicles, aerial vehicles such as UAVs/drones bring new possibilities for IoT applications. In the following examples, we explore how they have been utilized to improve both the sensing and networking capabilities of IoT.

### 2.4.1 Mobility-Enhanced IoT Sensing with UAV and Drones

UAVs/drones, with their flexible and fast mobility, can carry sensors to provide valuable real-time aerial perspectives, offering unique data insights from above. Moreover, they have the capability to reach extreme conditions and locations that are otherwise challenging or inaccessible for traditional ground-based approaches, making them indispensable assets in diverse IoT applications. Example usage of them includes building inspection [80, 177] where drones mounted with sensors such as a ladder, RGB, or infrared cameras provide a cost-effective and safe way for 3D-reconstruction or thermal profiling of buildings.

Drones have also been used to assist in environment monitoring. For example, authors in [64] explored the approaches for motion planning of drones for capturing aerial imagery and image processing techniques for post-disaster assessment and monitoring of infrastructure development. UAVs have also found extensive applications in real-time traffic monitoring and management [102]. They contribute to data collection for monitoring traffic congestion and roadway conditions, as well as detecting car accidents. Kim et al. [112] conducted an investigation on a framework that utilizes diverse small UAVs, along with public transit fleets, to facilitate time-sensitive monitoring of smart cities. Additionally, they explored the application of UAVs in monitoring ocean areas. Motlagh et al. [148] developed a UAV-based IoT platform for crowd surveillance, employing UAVs to collect aerial videos. They further explored the utilization of onboard or mobile edge computing for video processing, particularly for tasks such as face recognition.

Drone-based approaches have been extensively explored for enabling real-time disaster monitoring [58]. For instance, in an application for building fire localization [166], drones equipped with infrared sensors are dispatched to fly around building facades, scanning the walls to detect fire spots. Jimenez-Jimenez et al. [94] employed drones equipped with high-resolution cameras and image processing methods for objective detection, aiding real-time flood damage

assessment. Additionally, Schaefer et al. [191] presented a low-cost and rapid workflow for quantifying geomorphological changes in the aftermath of a natural disaster. In this work, UAVs were used to collect aerial imagery from hurricane-affected key sites in Dominica. In the context of wildfires, drones are utilized to capture RGB and thermal images for tracking the fire perimeter [77] and monitor fire intensity deviation [168] in real-time. Furthermore, a UAV-based tracking system [11] has been developed to create safe maps through disaster-stricken zones. Drones are intelligently planned to scan the disaster area, and aerial imagery is analyzed for localizing and tracking pedestrians in need of assistance. Additional in-depth discussions on the technical aspects of drones in disaster scenario monitoring can be found in Sections 5.2 and 6.2.

#### **2.4.2 Aerial Vehicles in Data Transmission**

Additionally, the mobility of aerial vehicles has also become instrumental in complementing data transmission between sensors and remote data centers. Equipped with sensors and communication technologies, drones efficiently collect data from IoT devices and facilitate its delivery to data analysis centers through wireless techniques such as WiFi, LTE, and 5G [12]. Authors in [149] investigated the deployment of multiple UAVs for data collection from ground IoT devices. They devised an approach to cluster IoT devices efficiently and optimized the motion of the UAVs to minimize power consumption for both data transmission and drone operations. Alsamhi et al. [13] explored the collaboration between drones and IoT devices for safety purposes, particularly to enhance recovery services after disasters in smart cities. They demonstrated how drones can be leveraged to provide timely wireless communication for improving network capacities in critical situations. Koulali et al. [119] conducted a study on cooperating with a group of drones to expand wireless communication coverage on demand. Their research focused on optimizing the beaconing period of each drone using game theory tools to minimize overall energy consumption. Drones were employed as data

mules in work [216] to gather data from ground sensors and deliver it to remote base stations, aiming to offer a cost-effective solution to enhance wireless communication. The authors of the study focused on optimizing the task allocation and path planning of drones to efficiently fulfill periodic data-gathering objectives.

## 2.5 Guiding Mobile Entities: Insights from Various Research Fields

The topic of planning and controlling the actions of mobile entities has been addressed in several research fields, including operations research (OR) [142], artificial intelligence (AI) [211], and robotics [50]. This interdisciplinary approach enables a comprehensive strategy to address the challenges of guiding and managing these entities.

The area of Operations Research employs mathematical models and optimization techniques to tackle intricate decision-making challenges, offering high-level strategic planning to guide mobile entities while accounting for complex application demands. OR research has developed techniques for route planning, task scheduling, and resource allocation of multiple mobile entities. These efforts often involve the utilization of techniques such as linear programming and (mixed) integer programming [115, 26, 121]. Additionally, a diverse array of heuristic strategies [125, 210] and advanced metaheuristic approaches, including tabu search [10, 32, 213] and evolutionary algorithms [161, 170], can be effectively employed. These techniques are aptly suited for resolving challenges like vehicle routing and multi-agent task assignment [142, 14].

AI also plays a significant role in decision-making processes and action planning for mobile entities in pursuit of application-specific objectives, facilitating adaptability in dynamic settings. Notable techniques encompass the utilization of the Planning Domain Description



Language (PDDL) with solvers like Distoplan [211, 212], Markov decision processes [152, 233], heuristic search methods [87, 205], and Monte-Carlo tree search [150, 179, 110] for multi-entity action planning. Additionally, machine learning methodologies [228, 103] encompassing reinforcement learning [229, 246, 44], and supervised and unsupervised learning [97, 89] come into play, enabling the development of algorithms enabling mobile entities to assimilate experience, perceive their surroundings, and execute informed decisions. In comparison to OR approaches, AI techniques exhibit enhanced adaptability to fluctuating environments, adeptness in data-driven decision-making, and real-time responsiveness, even within intricate and uncertain scenarios. However, it's important to note that AI's efficacy relies on accessible data, and certain techniques may lack transparency in decision-making processes.

Robotics provides the practical embodiment of plans, incorporates sensor data for context-awareness, and enables physical interaction and movement of mobile entities [244, 207]. Techniques from control theory [153, 60, 198], localization [71, 189, 74], mapping [62, 183, 72], and navigation [105, 192] play a crucial role in ensuring that mobile entities can move accurately and safely. Robotics primarily focuses on lower-level motion control, mainly addressing the physical execution of actions. However, when it comes to higher-level planning, robotics encounters challenges in effectively managing complex application requirements and optimizing long-term plans.

This thesis is dedicated to the enhancement of community IoT applications through the optimization of mobile entities, thereby improving sensing and networking capabilities. We specifically target time-sensitive aspects and devise novel planning and scheduling strategies tailored to various types of mobile devices, taking into account the distinct challenges and tradeoffs in diverse scenarios. To achieve these goals, we need to design middleware that aligns with the application's requirements and make decisions to guide mobile entities to fulfill these objectives. This process involves both higher-level planning and lower-level

control of mobile entities, incorporating approaches from the three research fields mentioned above, each of which presents unique challenges and considerations. For data collection and transmission, we must specify timing requirements, including data transmission delay tolerance, data collection frequencies and deadlines, and define spatial requirements, such as monitoring targets, areas, sensor needs, and data quality requirements for specific objectives. When planning and scheduling the actions of mobile entities, we incorporate methodologies from operations research and AI fields, enhancing them with novel considerations regarding spatial and temporal data collection requirements. This becomes especially critical in dynamic settings where repeated data collection is a necessity. Furthermore, we must carefully manage the trade-offs between sensor coverage and the quality of captured data, all while accounting for the possibility of network disconnectivity during data transmission. On a lower control level, we manage actions and movements using robotics-based techniques. In this thesis, we develop precise path-planning algorithms to ensure adequate data coverage and 3D obstacle avoidance. This thesis undertakes the challenge of addressing all the aforementioned complexities and interdisciplinary considerations, offering a comprehensive approach to enhancing IoT systems using mobility.

# Chapter 3

## Approach Overview

This chapter provides an overview of our proposed solution, which leverages planned mobility to enhance the sensing and network capabilities of community-based IoT applications, where time sensitivity is a crucial concern during data collection and analysis. In the following sections, we first describe the range of community IoT applications and the specific characteristics of time-sensitive IoT applications. Next, we highlight the reasons why utilizing planned mobility is well-suited for serving in these applications. Subsequently, we present an overview of our proposed approaches to integrate mobility into IoT systems in three main scenarios, catering to diverse time-sensitive IoT applications in urban and remote areas and addressing various challenges and tradeoffs.

### 3.1 Time-Sensitive Community IoT Applications

This thesis delves into the utilization of mobility to enhance time-sensitive community-based Internet of Things (IoT) applications. Time-sensitive IoT applications [118] demand timely completion of data collection and analysis to maintain data value and relevance. The term

”time-sensitive” underscores the significance of data freshness and accuracy, as any delay in data processing may result in a decay of its value. The time duration for data processing varies depending on the application’s requirements. Some applications necessitate real-time or near-real-time data collection and processing. For example, in emergency response systems for fire incidents [135], first responders rely on real-time information to track the dynamic status of the fire situation and promptly execute human rescuing or firefighting tasks. Healthcare monitoring applications also necessitate real-time reporting of patients’ statuses to promptly notify the hospital. Another instance involves traffic management, where real-time data from sensors and cameras is used to monitor traffic flow and detect accidents. This immediate information enables the transportation department to adjust traffic signals correspondingly.

Conversely, certain time-sensitive IoT applications can tolerate longer delays but still require data processing within specific time durations based on their application requirements. For instance, in environmental monitoring applications, sensor data is collected for long-term trend analysis, such as noise assessment and air quality monitoring, allowing for relatively longer delays extending to hours or even days. Similarly, some applications aim to monitor traffic flow to analyze passengers’ daily planning, with data collection occurring daily or even over extended periods. In smart agriculture, data collection, analysis, and control can happen periodically, with intervals of a few hours, facilitating efficient crop monitoring and irrigation control for farmers. In these cases, while real-time data processing may not be critical, adhering to the specific time durations and delay tolerance remains essential to derive valuable insights and support decision-making processes.

## 3.2 Opportunistic vs. Planned Mobility

This thesis aims to explore the integration of mobility into three time-sensitive IoT applications, each dedicated to solving specific challenges. In this context, 'mobility' refers to the movement of physical entities such as humans, vehicles, and drones, which introduces new opportunities for IoT. These opportunities include achieving pervasive sensing coverage and implementing a cost-effective approach for data collection and transmission while also enhancing the flexibility and scalability of the IoT ecosystem. To investigate the approach to utilizing mobility, we begin by categorizing it into two distinct types: opportunistic and planned mobility, based on the extent to which the movement of these entities can be predicted or controlled.

### Opportunistic Mobility

The opportunistic mobility category encompasses mobile devices within the IoT that exhibit unplanned and unpredictable movement based on users' personal activity goals, such as crowds with smartphones, private cars, and animals. In this context, data collection or communication is triggered opportunistically whenever a mobile device comes into proximity with relevant phenomena or data sources. The related work in Section 2 highlights a wide array of IoT applications that make effective use of this particular mobility type. Examples include Crowdsensing-based applications [232] that rely on human mobility, as well as applications that leverage the mobility of personal cars [43] or bicycles [63]. In these applications, mobile entities opportunistically collect or transmit data.

It is important to note that opportunistic mobility is best suited for IoT scenarios that do not require time-critical data collection, control procedures, or stable and reliable data transmission. *However, for time-sensitive applications, where data collection and analysis must be accomplished within specific timeframes, opportunistic mobility may not be suitable due to its unpredictable and uncontrollable nature.*

## Planned Mobility: Predefined and On-demand Mobility

Next, we explore the mobility category that contrasts with opportunistic mobility, known as planned mobility. Under this pattern, the movement of mobile entities is predictable, providing a high level of certainty when assisting data collection and transmission tasks in IoT applications. Within this category, we go deeper to identify two types of mobility patterns: **predefined mobility** and **on-demand mobility**. The first type contains mobile devices that operate according to predefined schedules and trajectories. This includes public transit systems such as buses, city fleets, and railways. While these devices have constraints on their activity, they bring certainty and predictability to their mobility patterns. Leveraging this mobility can provide more reliable and stable sensing procedures and data transmission in IoT applications. However, utilizing this mobility type for sensing and data transmission requires considering its constrained activity range. Events monitored by these mobile entities must occur near their trajectories, and data source and network access points must be located along their route. An illustrative example of an application that leverages this mobility is smart transportation systems [73]. In such systems, mobile devices like buses or city fleets are equipped with sensors to monitor transportation information and environmental conditions along the bus routes. Additionally, trains are utilized in other applications [238] to provide periodic data delivery, collecting media-rich data from local data sources and transmitting it to remote data processing centers along the railway routes.

The category of on-demand mobility revolves around the utilization of mobile devices in IoT applications that promptly respond to specific tasks initiated by users and are dispatched to locations where events of interest occur for monitoring or assisting with data transmission. This category encompasses devices such as drones, robots, or unmanned vehicles, whose movement can be flexibly controlled by users. Examples of applications in this category include drone-based surveillance applications [64, 135], where drones equipped with sensors fly to disaster settings to provide real-time sensor data for monitoring dynamic environ-

ments. Additionally, in other applications [248], bicycles are dispatched to support mobile sensing and data collection in community-based IoT applications for environmental monitoring. Overall, on-demand mobility offers more flexibility and scalability to IoT systems compared with predefined mobility, with fewer limitations on device movement. However, the effective utilization of mobile entities requires careful consideration of their deployment, motion planning, and scheduling for sensing and networking assistance.

Figure 3.1 illustrates the characteristics and exemplary mobile entities under opportunistic mobility and the two types of planned mobility.

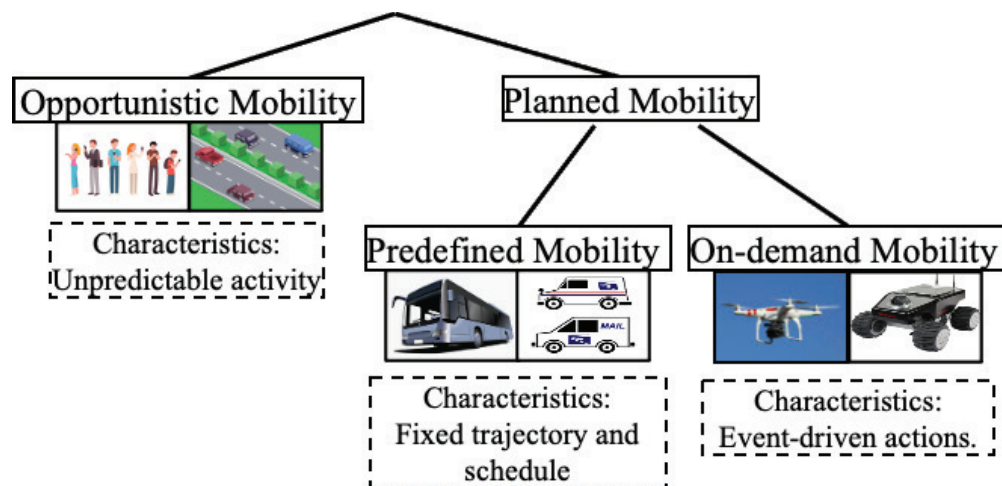


Figure 3.1: Categories of Mobility

Based on the above description, it is evident that utilizing planned mobility in mobile sensing or data transmission provides the ability to plan and control the actions and movement of these entities for efficient data collection and transmission while considering the specific delay tolerance requirements of the application.

### 3.3 Leveraging Planned Mobility to Enhance IoT

This thesis aims to integrate mobile entities with planned mobility patterns into IoT systems, enhancing their sensing and networking capabilities in time-sensitive applications. To

achieve this, we first characterize the data flow within an IoT system with mobile devices in three layers: the physical world, middleware layer, and application layer (as shown in Figure 3.2). In this system, in-situ sensors and mobile devices equipped with sensors and network communication tools continuously monitor the physical world, collecting sensor data and transmitting it to the middleware layer, which may be a local server or a remote cloud platform, through various networks. Mobile devices also can play a critical role in IoT systems by serving as data carriers (mules) to complement network infrastructures and contribute to cost-effective and seamless data transmission in IoT ecosystems.

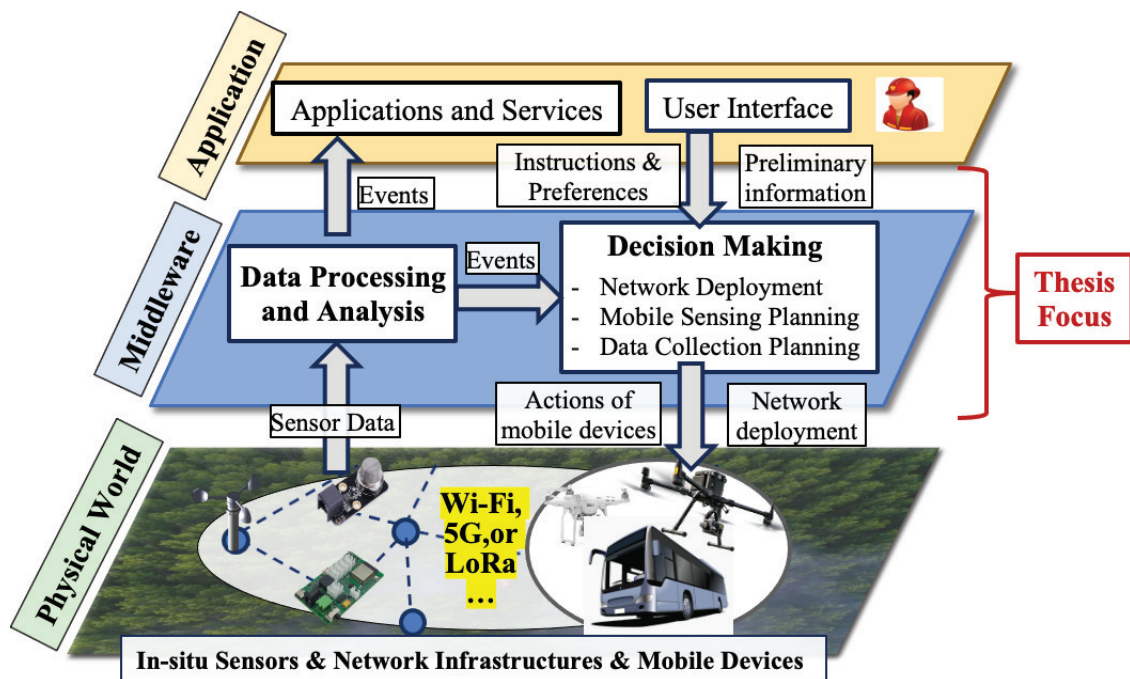


Figure 3.2: IoT System Architecture with Selected Components

Within the middleware layer, the main focus of this thesis, various computing components process and analyze data, extracting meaningful “events” for different application purposes. These events are then reported to the application layer, where users can utilize them for specific business, city management, or industrial control needs. Additionally, certain decision-making components in the middleware layer are dedicated to planning and controlling the deployment of mobile entities and network infrastructures. Their goal is to execute tasks related to sensing and data transmission based on user-specific instructions and preliminary



information.

This thesis explores strategies for planning mobile devices to enhance sensing and network capabilities in three diverse scenarios. These scenarios cover a wide range of community-based IoT applications, from urban to remote areas, each presenting unique time-sensitivity requirements and design challenges. Next, we will describe the three driving-use case scenarios and summarize our designed IoT system and proposed novel approaches in the middleware layer for each scenario. As we progress from one scenario to another, the level of system's autonomy increases, resulting in a more automatic and efficient deployment and operation.

### **3.3.1 Scenario 1: Mobility for Cost-Effective Network Coverage**

We first look into IoT applications in urban scenarios characterized by high population density and extensive sensor deployment for monitoring various conditions. A challenge faced in these scenarios is the limitation of local network resources to handle long-range data transmission. The increasing number of IoT devices equipped with media-rich sensors, such as cameras, generates massive volumes of data, placing considerable strain on existing network infrastructure. While long-range data transmission techniques such as cellular networks and 5G can provide sufficient bandwidth, they often come with high installation and maintenance costs. On the other hand, cost-effective alternatives like LoRa [88] and Sigfox [126] may not be well-suited to handle the large data volumes associated with media-rich sensors. Therefore, it becomes crucial to explore innovative solutions that support effective data transmission and network usage, ensuring a cost-effective and sustainable IoT implementation in urban scenarios.

In this context, we focus on applications with relatively delay-tolerant time sensitivity for sensor data collection and transmission. The data collected by sensors in such scenarios is typically not immediately critical, allowing for periodic data collection and transmission

at predefined intervals. For instance, environmental sensors may measure air quality levels every hour, making data aggregation and periodic transmission sufficient for their purposes. Notably, the time sensitivity requirements can vary across different IoT applications. For instance, healthcare monitoring applications may demand low-latency data transmission to provide timely and critical updates to healthcare providers. On the other hand, other IoT applications in urban environments, such as those related to urban planning and resource management, can handle delays of minutes, hours, or even longer without compromising their effectiveness. Above all, in this scenario, the critical challenge is to design an effective and cost-efficient solution for timely data transmission from pervasive sensors, considering their diverse delay tolerances, all while minimizing the strain on local networks.

To address these challenges, we propose utilizing predefined mobility, specifically public transit fleets, to assist in data transmission and complement the constrained network resources. The wide coverage and predictability of the mobility pattern of public transit fleets make them suitable for this purpose. In planning the utilization of predefined mobility, we plan to install network infrastructure along the routes of the public transit fleets and devise a data collection and upload process for these mobile devices. Key considerations in this design include the heterogeneous nature of delay-tolerant sensing data and the cost-effectiveness of the overall solution.

### **3.3.2 Scenario 2: Utilizing Mobility to Enhance Sensing Coverage**

In the second scenario, our primary focus is on utilizing IoT applications for emergency response in urban areas. In these scenarios, the IoT system is designed to continuously monitor the dynamic conditions of disaster situations, which are also regarded as emergent events due to their unpredictable and critical nature. These IoT applications aim to greatly enhance situational awareness for first responders during disaster scenarios, enabling them

to make rapid and accurate decisions. As a specific use case, we delve into high-rise fire monitoring. In this scenario, firefighters require real-time sensing data, including RGB and thermal images, to continuously monitor various aspects of the fire situation, such as the spread of the fire, the locations of individuals, and the status of building ventilation.

One of the significant challenges in deploying IoT in emergency situations like high-rise fires is the extreme conditions that can impede sensing and networking capabilities. For example, in high-rise fire settings, the height of the buildings and the extent of fire damage can result in limited sensor coverage. Some areas may become inaccessible to humans, making it difficult to deploy sensors effectively and gather critical data from those regions. Additionally, the severity of the fire can cause damage to sensors and network infrastructures located in the buildings, rendering them non-functional. Furthermore, monitoring dynamic and unpredictable emergent events requires first responders to have instant access to the evolving environment and continuously track its evolution. This necessitates real-time (or near real-time) sensor data collection and continuous (or periodic) monitoring. Additionally, in this scenario, first responders always need to track multiple objectives, including humans, fire, smoke, and building conditions. Each objective has unique monitoring requirements based on its dynamics, urgency, and importance. During the data sensor data collection process monitoring, it is crucial to consider the varying priority needs of these objectives to enable better resource allocation, ensuring the appropriate direction of resources toward the most critical objectives.

With the above concerns, we propose to integrate event-driven mobility, specifically drones, into the IoT system in this scenario, considering their fast and flexible mobility in the sky. Drones can navigate complex building structures, capture data from different perspectives, and provide real-time data transmission. We design an IoT platform that serves as an interface for firefighters to input monitoring tasks for drones, a dashboard to represent the dynamic fire situation and coordination mechanisms for multiple drones to fulfill monitoring

tasks. There are several concerns related to drone flight planning in this scenario. These include task allocation among multiple drones, coordination of drone movements, and obstacle avoidance during flight. The design of the IoT system needs to address the scheduling and planning of drone motions to monitor specific areas in a periodic manner, taking into account the heterogeneity of monitoring periods and priorities. Moreover, during the monitoring process, we strategically position drones at diverse distances from the building facade to achieve varied sensor coverage and data quality of captured data, such as image spatial resolution. Closer proximity to the monitoring objective results in higher data quality, but it may limit sensor coverage. When planning drone flight, we carefully consider this trade-off between maximizing sensing coverage and improving data quality.

### **3.3.3 Scenario 3: Mobility for Enhanced Networking and Sensing**

In the third scenario, we delve into monitoring emergent events in remote areas, where additional challenges arise due to the absence of network infrastructures and poor network conditions. Using wildland fire monitoring as a driving use case, the primary objective is to achieve continuous monitoring of dynamic fire settings in forest areas where there is a lack of cellular network infrastructures. Even the deployed local wireless networks face poor network conditions due to signal attenuation caused by blocking trees or other obstacles. In addition, the sporadic and unpredictable nature of wildland fires or other events in large areas makes it impractical and cost-effective to deploy sensors before such events occur. Moreover, these fires can cause damage to in-situ sensors, rendering them unavailable for monitoring. As a result, the lack of available sensors and network infrastructures in these scenarios poses a critical challenge for IoT deployment to support continuous monitoring and real-time data collection. Furthermore, in large-scale and long-term monitoring processes, it is crucial to consider automation for monitoring operations when designing IoT systems. Minimizing human involvement and automating the monitoring system also become vital aspects in this

scenario.

In this context, event-driven mobility remains crucial for real-time data collection and transmission during the monitoring of emergent events, given the lack of network infrastructure. We leverage the flexibility and adaptability of event-driven mobility to facilitate sensor data collection and transmission. To meet diverse application requirements, drones equipped with various sensors, such as RGB and thermal cameras, are utilized for multiple monitoring purposes, including fire tracking, fire intensity inspections, and human detection. Automation of the monitoring process is achieved through a Cyber-Physical Systems (CPS) control loop.

To implement this control loop, we employed data analysis approaches and designed semantics to represent the physical world. Additionally, we developed an automatic task generation procedure for generating tasks for drones and designed a multi-drone flight planning approach to guide drones in fulfilling these generated tasks. During task generation, we used a physical model to predict fire evolution, enabling prioritization of monitoring areas. This optimization allows us to make the most of drone resources and improve monitoring efficiency during emergencies. For example, tasks can focus on areas where the fire is expected to spread rapidly, while fireproof areas may not require fire detection tasks. Additionally, in the design of multiple-drone flight planning approach, we take into account the heterogeneity of drones and sensor configurations, the trade-off between sensor coverage and data quality, and the time-sensitivity of monitoring diverse objectives, which have varying monitoring periods and priorities. Furthermore, we address the potential network disconnection during the flight planning procedure. Specifically, we enable drones to store data onboard when disconnected from the ground edge server and upload it once the network connection is reestablished. This approach facilitates drone-assisted data ferrying and enhances the overall data transmission process in such situations.

Figure 3.3 summarizes the challenges of IoT applications in these three scenarios and provides an overview of the corresponding solutions.







Use Case	Challenges	Mobility	Approach	Research Problem	Special Concerns
<b>Smart Cities</b> 	Limited local network resources	 <b>Predefined (Buses)</b>	Vehicles ferrying sensor data to remote data center	<ul style="list-style-type: none"> <li>- Network deployment</li> <li>- Data ferry planning</li> </ul>	<ul style="list-style-type: none"> <li>-Diverse data delay tolerance and priority</li> <li>-Cost limit</li> </ul>
<b>High-rise Fire Monitoring</b> 	Unavailability of sensors	 <b>On-demand (Drones)</b>	Drones aiding mobile sensing	<ul style="list-style-type: none"> <li>- System design</li> <li>- Multi-drone flight planning</li> </ul>	<ul style="list-style-type: none"> <li>-Continuous monitoring</li> <li>-Diverse monitoring objectives</li> <li>-Coverage vs accuracy</li> <li>-Obstacle avoidance</li> </ul>
<b>Wildland Fire Monitoring</b> 	Unavailability of sensor and network infrastructure	 <b>On-demand (Drones)</b>	Drones aiding sensing and network	<ul style="list-style-type: none"> <li>- System design</li> <li>- Task generation</li> <li>- Multi-drone flight planning</li> </ul>	<ul style="list-style-type: none"> <li>-All in Scenario 2,</li> <li>- System autonomy</li> <li>-Network disconnection</li> <li>-Heterogenous drones and sensors</li> </ul>

Figure 3.3: Challenges and Solutions for IoT Applications in Three Explored Scenarios

# Chapter 4

## Cost-Effective Data Transmission with Public Transportation Fleets

This chapter addresses the utilization of predefined mobility to tackle network coverage challenges in IoT systems within urban areas. We use smart cities as the driving use case, where tremendous media-rich sensors are deployed pervasively; this poses significant challenges to local access networks that often have limited bandwidth. To address this, we investigate how to leverage public transit fleets (i.e., city buses) as a cost-effective solution for long-range data transmission to complement limited local networks. Here, we introduce the concept of “IoT zones” to capture geographically correlated clusters of local IoT devices with well-connected wireless networks that may have limited access to the Internet. We then develop techniques to create a cost-effective data collection network using existing transportation fleets with predefined schedules to collect sensor data from IoT zones and upload them at locations with better network connectivity. Such techniques will address tradeoffs between collection quality, timing needs (QoS), and network infrastructure installation costs.

## 4.1 Chapter Overview

Internet-of-things (IoT) deployments are giving rise to smart cities and communities worldwide. The next generation of urban planning is moving towards the design of smart instrumented spaces beneficial to citizens. For example, the City of Barcelona introduced the concept of *superblocks* to limit transportation within cityblocks [181]. A large number of interesting urban smart greenspaces efforts are being initiated in cities throughout the US [165], the goal is to support increased environmental sustainability and improved quality of life for citizens. Studies [180] also point out that the market for short-range IoT wireless technologies (Wi-Fi, BlueTooth, Zigbee) have overtaken those for long-range connectivity (LoRa, Sigfox), which can probably be attributed to the lower hardware and spectrum costs associated with the short-range networks. In the future, we envision that such innovative designs will give rise to localized **IoT zones**, where geographically-correlated clusters IoT devices are interconnected via various wireless networks (short and long-range).

Modern IoT devices are often equipped with media-rich sensors, such as microphones, cameras, and Radar/Lidar sensors, which generate tremendous volumes of sensor data. Data obtained from these devices must be fused, analyzed, and interpreted – solutions that require onboard storage, computation, and communication for complex analytics in every device are prohibitively expensive. The ability to create local collection points is plausible within a zone using M2M technologies and simple analytics can be executed at these network edges. Enabling high bandwidth networks to backhaul data at the device level to big data processing backends for more comprehensive analysis is both expensive and difficult. Communications of large IoT sensor data, such as surveillance camera footage (for further analysis or archival) or real-time social media sharing, is challenging since access networks usually have limited bandwidth and are vulnerable to network congestion. This work deals with the problem of getting data from local edges to the backend in a cost-effective manner.



Our proposed solution is to create a hierarchy where end-devices communicate to a local node that we refer to as a **Rendezvous Point (RP)**. Creation of such local edge components or RPs is becoming increasingly possible, e.g., through smart streetlights, smart transit stops, etc. Data make their way from the local RP to a larger processing backend through intermediate **Upload Points (UPs)** located at suitable places where better network connectivity and bandwidths are available. Two key questions need to be answered: (a) Where should Upload Points (UPs) be located (*upload point placement problem*) and (b) How do “data to be uploaded” get from local RPs to UPs (*upload path planning problem*)?

Our key intuition is to exploit existing transport possibilities to implement this hierarchy, i.e., by leveraging city transit infrastructures that currently transport people to transport data from devices to a big data processing backend. Such traffic fleets with predefined schedules (routes and times) are common in urban communities today, which include buses, mail vehicles, and garbage trucks with regular schedules and stops. We plan to leverage fixed city transit infrastructures, in particular, bus stops (or traffic lights), as potential RPs and UPs in our hierarchy. Since existing transportation fleets (e.g., city buses) have predefined schedules and paths, we will design solutions where data gathering (from RPs), transport (to UPs), and upload (from UPs to backend) can be made to occur along these paths. A rational solution is to colocate RPs and UPs with transit stops because vehicles are typically required to stop at these points, providing the needed time for reliable wireless transmission of data to/from vehicles.

The following are key contributions of this work.

- We develop an integrated approach to address the problems of *upload point placement* and *upload path planning* (Sec. 4.2).
- We develop a modeling framework and formulate the upload point placement problem, which is NP-hard (Sec. 4.3).

- We propose four upload point placement algorithms for optimized upload point deployment considering the trades of the spatial coverage, upload deadlines, and deployment feasibility for the target IoT zones under an installation cost limit. (Sec. 4.4).
- We develop two upload path planning algorithms in which Delay Minimization (DM) algorithm utilizes the upload point placement map generated in the previous step for minimizing delay, given an IoT deployment with data generation patterns, communication needs, and vehicle schedules (Sec. 4.4).
- We validate our approaches and compare them using real-world transportation networks—i.e. road map and transit schedules for Orange County, CA (Sec. 4.5).

## 4.2 Sample Scenario and Problem Statements

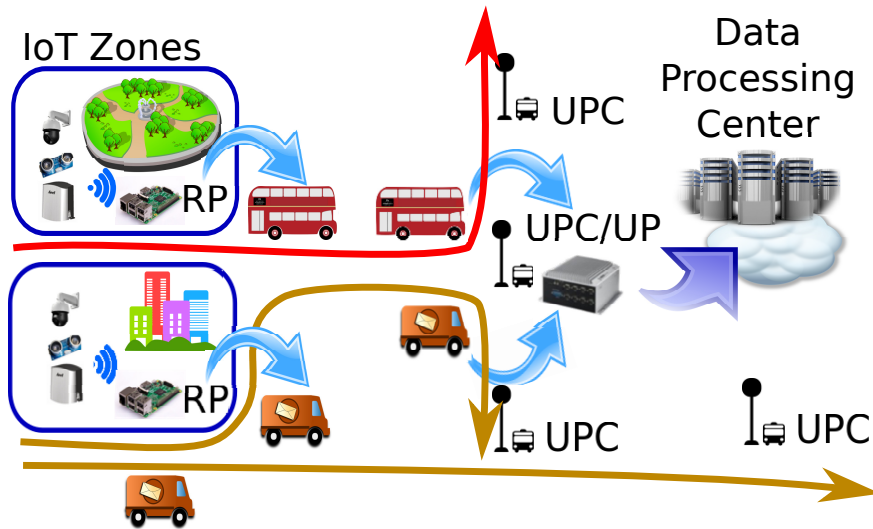


Figure 4.1: Sensor data collection with scheduled fleets.

Fig. 4.1 is a simplified usage scenario with two IoT zones (park, business district) each with devices that are well connected, and communicate sensor data to local RPs. RPs possess sufficient storage to buffer incoming sensor data until a scheduled pickup vehicle on a trip arrives. Sensor data items are associated with a delay tolerance, i.e. a tolerable time

difference between arriving at RPs and at a data processing center. Each vehicle contains a computing device with storage; vehicles in transit pick up data buffered at RPs and transport them to an *Upload Point* (UP) from where sensor data is delivered to the data processing center. Differently from RPs, whose locations are given, UPs can be set up at various locations, such as bus stops, road intersections, street lamps, and roadside buildings. We refer to the possible locations for setting up UPs as *Upload Point Candidates* (UPCs). Every UPC has its own *installation cost*, which depends on practical factors, such as type of access network technologies, distance to Internet access points, and availability of power supply. We address two core research problems in this setting:

- *Upload point placement.* Selection of a subset of UPs from all possible UPCs to ensure spatial coverage and satisfy delay tolerance, under a budget of UP installation cost.
- *Upload path planning.* Plan for each vehicle and RP that schedules for pick-up and drop-off of data gathered at RPs, given the transportation fleet schedules.

In Fig. 4.1, among the three potential UPCs, the center candidate maximizes the number of RPs served with lower delays and are hence selected as an upload point. In practice, the choice of an upload point will take into consideration the installation costs of individual UPCs and transit vehicle paths and schedules. Several usage scenarios of this infrastructure can be envisioned. Consider public parks instrumented with surveillance cameras and environmental sensors that communicate data to a local collection point; data are transported to suitable upload points for further analysis to provide live situational awareness of public spaces (e.g., occupancy), detect public safety threats (e.g., fires, protests) or to promote community recreational events. Similarly, residents in assisted living facilities can send personal health and community activity-related information to data processing centers via mail/carrier trucks.

**Existing Efforts in Predefined Mobility for Data Transmission:** We next supple-

ment the related work discussed in Section 2.3 by providing a more in-depth examination of research efforts for using ground vehicles to support delay-tolerant networking (DTN), wireless mesh networks, vehicular networks, and community-scale data collection. Clustering formations and energy-efficient resource allocation methods for wireless IoT devices have been explored in [214, 133]; DTNs and data mules have exploited mobility for data transfer [30, 243, 245]; these approaches often assume very limited access to communication infrastructure and aim to meet data transfer deadlines using multi-hop networks [111]. Techniques for proactive and adaptive data transmission in this setting have been designed with crowdsensing applications and energy efficiency [123, 230, 137, 173, 167]. Similar work in the context of WSN makes use of mobile sinks [117, 108, 120] for improved communication. Techniques to exploit public bus transportation for DTN-based data dissemination include utilizing bus line patterns [195], accounting for encounter frequency of bus routes [131] and using bus stops as communication relays.

Planning and deployment are often network-specific; techniques for AP and gateway placement in wireless mesh networks [227, 226] are useful for community-scale networking. Modeling using set cover based formulation [33], and techniques for network deployment [51] and performance [162] have been explored. Facility placement mechanisms for data upload in delay-tolerant crowdsensing and content distribution have been formulated and studied [201, 25]. Combining facility placement with transport logistics [176] for integrated provisioning is an approach similar to ours, albeit for goods transport. Related literature from the vehicular networking community includes work on the deployment and operation of vehicular networks [231], their graph properties [172], and methods for more realistic modeling of mobility [38, 107, 106]. More recent efforts have explored the use of public transport to collect IoT data in cities [99, 92]. Other related efforts include approaches to utilize Wi-Fi-enabled buses for non-urgent communications [6] and geocast-based mechanisms to improve delivery reliability and timing [240, 241] and monitor the urban environment [100].

Much of the earlier literature uses statistical estimates of traffic flow and encounter probabilities. In contrast, we leverage the knowledge of exact routes and transit schedules (especially in urban settings) to generate data routing plans and handle the heterogeneity of IoT traffic while satisfying communication QoS needs.

## 4.3 Problem Formulation

In this section, we formulate our upload point placement and upload path planning into one combined formulation.

### 4.3.1 Symbols and Notations

The transportation/transit fleet is described by a set of *trips*,  $\mathbf{V}$  with  $|\mathbf{V}| = V$ , where each trip  $v$  denotes a bus/vehicle moving through a sequence of *stop points* (i.e., bus stops). When a vehicle reaches its terminal stop and starts again, it is considered as a different trip. Let  $\mathbf{N}$ , with  $|\mathbf{N}| = N$ , be the set of all the  $N$  stops that any of these trips go by. We assume all data gets accumulated in these stop points and also gets uploaded via them. Thus set  $\mathbf{N}$  is the superset of all RPs and UPCs as well. We, in fact, can eliminate those stops from  $\mathbf{N}$  that are neither RPs nor UPCs because their presences do not affect our placement and planning. In that,  $\mathbf{N}$  becomes the set of all RPs and UPCs. For each stop point  $n$ , let  $R_n$  and  $D_n$  denote the data arrival rate and the data delay tolerance at  $n$ . Let  $C_n$  be the cost for installing a UP at  $n$ , respectively ( $R_n > 0$  and  $D_n > 0$  if  $n$  is an RP, or both equal to 0, and  $C_n = \infty$  if  $n$  is not a UPC).

We assume that the schedules of all trips are known apriori. Hence, we denote  $A_{n,v}$  as the arrival time of trip  $v$  at stop point  $n$  ( $A_{n,v} = -1$  if trip  $v$  does not go by  $n$ ). The schedule matrix,  $\mathbf{A} = [A_{n,v}]$  contains the complete schedule of all trips. The schedule holds for a

certain duration (e.g., for a day) and then perhaps repeats itself. We assume that IoT data collection happens sometime within this interval and let  $T_s$  and  $T_e$  be the start and end time of this data collection interval.

For the *upload point placement* problem, we define a binary array  $\mathbf{Y} = \{y_1, y_2, \dots, y_N\}$  to denote UP placement. In particular, stop point  $m \in \mathbf{N}$  is chosen as a UP iff  $y_m = 1$ . For the *upload path planning* problem, we define a three-dimensional auxiliary matrix  $\mathbf{X}$  of size  $N \times (V + 1) \times N$ , where  $x_{n,v,m} = 1$  iff trip  $v \in \mathbf{V}$  is used to carry data from stop point (RP)  $n$  to another stop point (UP)  $m$ ; and  $x_{n,v,m} = 0$  otherwise. Moreover, we use index  $v = 0$  to capture the corner cases where an RP/UPC is selected as a UP. In that case, the sensor data are directly uploaded by the UP whenever they reach the RP. Concretely, we let  $x_{n,0,m} = 1$  iff  $n = m$  and  $y_m = 1$ ;  $x_{n,0,m} = 0$  otherwise. In the following, if not otherwise stated, we use  $n$  to denote an RP stop point,  $v$  for a trip, and  $m$  for a UPC stop point.

We note that  $m \in \mathbf{N}$  should be selected as a UP in  $\mathbf{Y}$ , if at least one vehicle trip decides to dump sensor data at  $m$  in  $\mathbf{X}$ . This is,  $y_m = 1$  if and only if there exists at least one  $x_{n,v,m} = 1$  for some  $n$  and  $v$ . Moreover, each vehicle trip that picks up data at  $n$  uploads the data at a single UP. That is,  $\sum_m x_{n,v,m} \in \{0, 1\}$  for each  $n$  and  $v$ .

**Data Transfer Time:** As data chunks are moved from an RP to a UP via a trip, the chunk experiences some delay. For each  $x_{n,v,m} = 1$ , there is an associated data transfer time,  $d_{n,v,m}$ , that denotes the amount of time it takes to transfer data from RP  $n$  to UP  $m$  via trip  $v$ . The transfer time is a function of  $\mathbf{X}$  and it has two parts: wait time (waiting for the bus to arrive at the RP) and travel time (the time to reach the planned UP). The wait time depends on when was the last time data was carried by any trip passing by this stop point (since then, the RP is waiting for a bus to show up). Let  $B_{n,v}(\mathbf{X})$  be the last pick up time

preceding trip  $v$ . This is:

$$B_{n,v}(\mathbf{X}) = \max\{A_{n,v'} \sum_m x_{n,v',m} | A_{n,v'} < A_{n,v}\}. \quad (4.1)$$

If no preceding trip exists,  $B_{n,v}$  is set to  $T_s$ . So, the wait time becomes  $A_{n,v} - B_{n,v}(\mathbf{X})$ . Once loaded into bus trip  $v$ , the travel time to each UP  $m$  is given by:  $A_{m,v} - A_{n,v}$ . Adding the above two terms and canceling the common term, we obtained the data transfer time as follows:

$$d_{n,v,m}(\mathbf{X}) = A_{m,v} - B_{n,v}(\mathbf{X}). \quad (4.2)$$

**Data volume:** The volume of data carried by each vehicle trip affects the decisions, in the sense that the on-time delivery of a vehicle trip carrying more data should be more critical. We let  $S_{n,v}(\mathbf{X})$  be the data volume carried by vehicle trip  $v$  from stop point  $n$ , which can be written as:

$$S_{n,v}(\mathbf{X}) = R_n [\min(T_e, \max(A_{n,v}, T_s)) - \min(T_e, B_{n,v}(\mathbf{X}))] \sum_m x_{n,v,m}.$$

Here,  $\min(T_e, \max(A_{n,v}, T_s))$  and  $\min(T_e, B_{n,v}(\mathbf{X}))$  represent the arrival time of the last and first sensor data bits that are buffered at RP  $n$ , which will be picked up by vehicle trip  $v$ . The right-most summation is a binary value indicating if trip  $v$  picks up sensor data at stop point  $n$  or not.

**Penalty function:** The overall objective of data collection and transfer is to upload as much data as possible with lower transfer delay. Hence, we introduce the notion of a *penalty* function that accounts for both the volume of data as well as the delay the transfer experiences. That is, the data chunks transferred with larger delays incur more penalty than the ones that are transferred with smaller delays. There is also a penalty for data being not

uploaded at all at the end of the operation (after  $T_e$ ). Therefore, the overall penalty, denoted as  $\hat{P}(\mathbf{X})$ , measures how good a certain upload plan  $X$  is and thereby quantifies the service quality of sensor data collection. The penalty function is the sum of the following two terms:

*Penalty due to transfer delay:* This part measures the total accumulated transfer delay weighted by the respective volume of data, which is defined as:

$$P_l(\mathbf{X}) = \frac{1}{V} \sum_{n,v,m} f_n(d_{n,v,m}(\mathbf{X})) \cdot S_{n,v}(\mathbf{X}) \cdot x_{n,v,m}, \quad (4.3)$$

where  $V = (T_e - T_s) \sum_n R_n$  is the total volume of data generated and  $f_n(d)$  is used to normalize delay within  $[0, 1]$ , which is defined as  $f_n(d) = 1 - \exp\left(\frac{-d}{D_n}\right)^3$  with  $f_n(0) = 0$  and  $f_n(\infty) \rightarrow 1$ .

*Penalty due to data not being uploaded:* This part accounts for sensor data being left on RPs beyond the end of each schedule (say, overnight). Let  $L_n$  be the last time (within  $T_e$ ) when data is carried from stop point  $n$  by any trip. That is,  $L_n(\mathbf{X}) = \max\{A_{n,v} \sum_m x_{n,v,m}\}$ . We have:

$$P_u(\mathbf{X}) = \frac{1}{V} \sum_n R_n \cdot (T_e - L_n(\mathbf{X})) \cdot (1 - x_{n,0,n}). \quad (4.4)$$

The term  $(1 - x_{n,0,n})$  takes care of the corner cases where an RP is also chosen as a UP.



### 4.3.2 Formulation

We write the upload point placement and upload path planning problem (finding  $\mathbf{Y}$  and  $\mathbf{X}$  simultaneously) as follows:

$$\min \quad \hat{P}(\mathbf{X}) = P_l(\mathbf{X}) + P_u(\mathbf{X}) \quad (4.5a)$$

$$\text{s.t.} \quad x_{n,v,m} \leq y_m, \forall n \in \mathbf{N}, v \in \mathbf{V} \cup \{0\}, m \in \mathbf{N}; \quad (4.5b)$$

$$A_{n,v}x_{n,v,m} \geq 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \quad (4.5c)$$

$$A_{m,v}x_{n,v,m} \geq 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \quad (4.5d)$$

$$(A_{m,v} - A_{n,v})x_{n,v,m} \geq 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \quad (4.5e)$$

$$\sum_{m \in \mathbf{N}} x_{n,v,m} \leq 1, \forall n \in \mathbf{N}, v \in \mathbf{V}; \quad (4.5f)$$

$$\sum_{m \in \mathbf{N}} y_m C_m \leq \Theta; \quad (4.5g)$$

$$x_{n,v,m} \text{ and } y_n \in \{0, 1\}, \forall n \in \mathbf{N}, v \in \mathbf{V} \cup \{0\}, m \in \mathbf{N}. \quad (4.5h)$$

The objective function in Eq. (4.5a) is to minimize the penalty function value. The constraints in Eq. (4.5b) connect  $\mathbf{Y}$  with  $\mathbf{X}$ , so that  $y_m = 1$  if *at least* a vehicle trip  $v \in \mathbf{V} \cup \{0\}$  is determined to pick up sensor data at  $n$ , i.e.,  $x_{n,v,m} = 1$ ;  $y_m = 0$  otherwise. The constraints in Eqs. (4.5c) and (4.5d) prevent any vehicle trip  $v$  that doesn't pass stop point  $n$  ( $A_{n,v} = -1$ ) from retrieving sensor data from  $n$ . The constraints in Eq. (4.5e) guarantee that vehicle trip  $v$  always picks up sensor data before dropping them off. Eq. (4.5f) makes sure that if RP  $n$  sends data to vehicle trip  $v$ ,  $v$  only drops off the data at a single UP. Eq. (4.5g) caps the UP installation cost at  $\Theta$ , which is an input.

Our problem is NP-hard, which can be shown through a polynomial-time reduction from the knapsack problem [104] to it. The knapsack problem aims to pick items, each with an associated profit and a weight, to get the maximal total profit subjects to the total weight

limitation. We can map weight limitation to our cost limitation  $\Theta$ , items to our UPCs, and total profit to the negative of our penalty value, although our problem is more comprehensive, e.g., the benefit of choosing a UPC as UP is not just a fixed profit, which is impacted by the trip assignments denoted by  $\mathbf{X}$  and even other chosen UPCs. Details of the proof are omitted due to the space limitation.

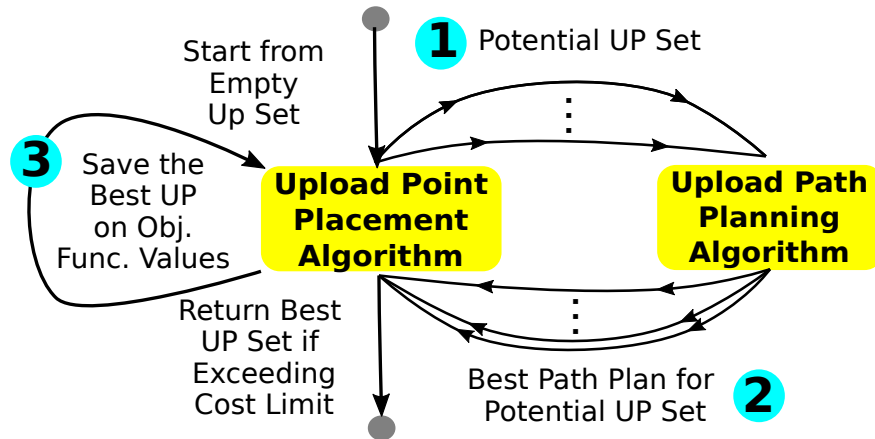


Figure 4.2: Our approach with two collaborating algorithms.

## 4.4 Solution Approach and Algorithms

We propose to iteratively solve the problems using two alternative algorithms:

- *Upload point placement algorithm*, which produces the upload point placement  $\mathbf{Y}$  based on multiple invocations of the next algorithm.
- *Upload path planning algorithm*, which is invoked by the above algorithm to compute the upload path plan  $\mathbf{X}$  for a given UP set  $\mathbf{Y}$ .

Fig. 4.2 illustrates the interactions between these two algorithms. The algorithm systematically *tries* different potential UP sets, which do not exceed the cost limitation. It only sees the high-level picture (set  $\mathbf{Y}$ ), and relies on the upload path planning algorithm to compute

---

**Algorithm 1:** Upload Point Selection Algorithm

---

**Input:** UPC set  $\mathbf{N}$ , schedule  $\mathbf{A}$ , rate  $\mathbf{R}$ , cost  $\mathbf{C}$ , delay  $\mathbf{D}$ , Cost limit  $\Theta$   
RP set  $\{n \mid n \in \mathbf{N}, R_n > 0\}$   
**Output:** UP placement  $\mathbf{Y}$  and upload path plan  $\mathbf{X}$

- 1  $\mathbf{Y} \leftarrow \{0\}_N; \mathbf{X} \leftarrow \{0\}_{N \times (V+1) \times N}$
- 2  $\mathbf{N}' = \{n \mid y_n = 0, C_n + \sum_{i \in N} C_i y_i \leq \Theta, n \in \mathbf{N}\}$
- 3 **while**  $\sum_{i \in N} C_i y_i < \Theta$  **and**  $\mathbf{N}' \neq \emptyset$  **do**
- 4     **for**  $n \in \mathbf{N}'$  **do**
- 5          $\mathbf{Y}'(n) \leftarrow \mathbf{Y}$ ; let  $y'_n \leftarrow 1$
- 6          $\mathbf{X}'(n) \leftarrow \text{Assign}(\mathbf{A}, \mathbf{Y}'(n))$
- 7         Calculate the cost-effectiveness for  $n$ :  $E(n) = \frac{\hat{P}(\mathbf{X}) - \hat{P}(\mathbf{X}'(n))}{C_n}$ .
- 8     **if**  $\max(E(n) \text{ for } n \in \mathbf{N}') == 0$  **then**
- 9         **break**
- 10      $i \leftarrow \arg \max_{n \in \mathbf{N}'} E(n)$
- 11      $\mathbf{Y} \leftarrow \mathbf{Y}'(i)$
- 12      $\mathbf{X} \leftarrow \mathbf{X}'(i)$
- 13      $\mathbf{N}' = \{n \mid y_n = 0, C_n + \sum_{i \in N} C_i y_i \leq \Theta, n \in \mathbf{N}\}$
- 14 **return**  $\mathbf{Y}$  and  $\mathbf{X}$

---

the low-level details (plan  $\mathbf{X}$ ). In particular, for each iteration, the upload point placement algorithm generates multiple potential UP sets and invokes the upload path planning algorithm multiple times (step ①). The upload path planning algorithm computes the best plan for each potential set (step ②). Toward the end of each iteration, the upload point placement algorithm selects the best UP set based on the objective function values (step ③). If the cost limit is exceeded (Eq. (4.5g)), the algorithms stop; otherwise, the upload point placement algorithm moves to the next iteration.

#### 4.4.1 Upload Point Placement Algorithms

We propose four upload point placement algorithms below. Intuitively, selecting UPCs with more passing trips from RPs will increase the chance of data uploads which in turn can decrease data loss and data transfer time. Based on this observation, we propose *coverage maximization* (COV) algorithm, in which we greedily select UPCs based on their coverage

of RPs. UP  $m$  is said to cover RP  $n$  if there are at least one trip  $v$  passing by  $n$  and arrives at  $m$  later, i.e., arrival time  $A_{n,v} < A_{m,v}$  and  $A_{n,v} \neq -1$ . According to the transit schedule  $\mathbf{A}$ , we get RP cover set  $\mathbf{Cov}_m$  for each UPC  $m$ . We then greedily choose the UPCs in the decreasing order of their  $\frac{|\mathbf{Cov}_m|}{C_m}$  until the cost exceeds the cost limit  $\Theta$ .

*Volume-maximization* (VOL) algorithm adopts the same heuristic method but uses the sum of the data rates of all those RPs instead of using only the count. The method then chooses UPCs in the decreasing order of volume to installation cost ratio until the cost hits the limit.

We also propose a *genetic algorithm* (GA) based solution where we create the initial populations using the solutions of COV and VOL. For each individual (a subset of UPC), we get its corresponding trip assignment using our **Assign** Algorithm and calculate the penalty value according to Eq. (4.3) and (4.4). We use the negative of penalty value as the fitness score to rank all populations. Then, we use three basic rules to create the next generation: (i) selection rules select a subset of individuals with highest fitness value as parents for the next generation, (ii) crossover rules combine two parents to form children, and (iii) mutation rules apply random changes to individual parents to form children. We also set the constraint of GA as the total cost of individuals shouldn't exceed the cost limit. In our work, we set the population size as 50 and the maximum number of iterations as 200.

The *upload point selection* (UPS) algorithm, as outlined in Algorithm 14, greedily finds UPs according to their reduction in penalty per unit of cost. More specifically, in each iteration, the algorithm computes the predicted upload path plan  $\mathbf{X}'(n)$  after adding each candidate  $n$  into the current UP set by using the subroutine **Assign**( $\mathbf{A}, \mathbf{Y}'(n)$ ) (line 6). It then calculates the cost-effectiveness of this assignment (line 7), which is the ratio between the decrease of the total setting penalty after adding  $n$  and the cost of  $n$ . We add the UPC that maximizes the cost-effectiveness to the UP set and go to the next iteration if the cost limit has not been reached.

---

**Algorithm 2:** Assign(**A**, **Y**) — DM algorithm

---

**Input:** Schedule **A**, UP placement **Y**, data rate **R****Output:** Upload Path Plan **X**.

```
/* Step 1: Adding all available trips */
1 X  $\leftarrow \{0\}_{N \times (V+1) \times N}$ 
2 for UP  $m \in \{m \mid y_m = 1\}$  do
3   for RP  $n \in \{n \in \mathbf{N}, R_n > 0\}$  do
4     if  $n == m$  then
5       | set  $x_{n,0,n} \leftarrow 1$ 
6     else
7       for  $v \in \{v \mid v \in \mathbf{V}, A_{m,v} \neq -1\}$  do
8         | if  $A_{n,v} \neq -1$  and  $A_{n,v} < A_{m,v}$  then
9           | | Set  $x_{n,v,m} \leftarrow 1$ 
10
11 /* Step 2: Trimming */
10 for RP  $n \in \{n \mid n \in \mathbf{N}, R_n > 0\}$  do
11   if  $x_{n,0,n} == 1$  then
12     | set  $x_{n,v,m} \leftarrow 0 \forall v \neq 0$ 
13   if  $\sum_{m \in \mathbf{N}} x_{n,v,m} > 1$  then
14     for  $m \in \{m \mid m \in \mathbf{N}, x_{n,v,m} = 1\}$  do
15       | if  $m = \arg \min_{m \in \mathbf{N}} A_{m,v}$  then
16         | |  $x_{n,v,m} \leftarrow 1$ 
17       else
18         | |  $x_{n,v,m} \leftarrow 0$ 
19
20 /* Step 3: Removing trips. */
19 for RP  $n \in \{n \mid n \in \mathbf{N}, R_n > 0\}$  do
20   Get all trips passing  $n$ :  $\mathbf{V}_n = \{v_{n,v,m} \mid \sum_m x_{n,v,m} = 1\}$ 
21   Sort  $\mathbf{V}_n = [v_n[1], v_n[2], \dots]$  by arrival time  $A_{n,v}$ 
22   for each trip in  $\mathbf{V}_n$ :  $v_n[i]$  do
23     | Get travel time  $t_n[i] = A_{m,v} - A_{n,v}$ 
24     | Get arrival time  $a_n[i] = A_{n,v}$ 
25    $updated \leftarrow True$ 
26   while  $updated$  do
27     |  $updated \leftarrow False$ 
28     for  $i = 1$  to  $len(\mathbf{V}_n) - 1$  do
29       | if  $t_n[i] > 2 \times (a_n[i+1] - a_n[i]) + t_n[i+1]$  then
30         | | Remove trip  $v_n[i]$ :  $x_{n,v,m} \leftarrow 0$ 
31         | |  $updated \leftarrow True$ 
32 return X
```

---

## 4.4.2 Upload Path Planning Algorithms

We propose two upload path planning algorithms: (i) *First Contact* (FC): every RP sends buffered data through all passing-by vehicles that then drop the data to the first UPs they encounter, and (ii) *Delay Minimization* (DM) algorithm, which performs the local search for optimal upload path plans. The algorithm works as follows (shown in Algorithm 2). For each RP, the algorithm produces the subset of trips that should carry data and upload them to the nearest UP they encounter. One can argue that an RP can send data through *all* passing trips and transfer a little chunk of data at each encounter. But it turns out that choosing all trips may not be the best, rather skipping some trips can generate better results (produce lower total transfer time/delay). Particularly, the trips that take a long travel time to reach their nearest UPs can be skipped. The following lemma establishes the condition.

**Lemma 1** (Removing Trips). *Let RP  $n$  see two successive trips  $v_i$  and  $v_{i+1}$  with the corresponding wait times as  $w_i$  and  $w_{i+1}$  and travel times to their respective nearest UPs as  $t_i$  and  $t_{i+1}$ . If  $t_i > 2 \times w_{i+1} + t_{i+1}$ , then trip  $v_i$  can be skipped which will decrease the overall penalty of transfer delay.*

*Proof.* According to Eq.(4.3) we can infer that the penalty of transfer delay is positively correlated to the weighted data transfer delay, and whether RP  $n$  chooses trip  $v_i$  to send data only impacts the transfer delay of the data arriving at RP  $n$  during wait time  $w_i$  and  $w_{i+1}$ . If RP  $n$  chooses to send data through both trips  $v_i$  and  $v_{i+1}$ , the former trip transfers  $w_i R_n$  volume of data with transfer delay  $w_i + t_i$  and the latter trip carries  $w_{i+1} R_n$  amount of data with delay  $w_{i+1} + t_{i+1}$ . So, the total weighted transfer delay of data arriving during  $w_i$  and  $w_{i+1}$  is  $w_i R_n \times (w_i + t_i) + w_{i+1} R_n \times (w_{i+1} + t_{i+1})$ . If the first trip is skipped then the sum becomes  $(w_i + w_{i+1}) \times R_n \times (w_i + w_{i+1} + t_{i+1})$ , which will be smaller than the former one if  $t_i > 2w_{i+1} + t_{i+1}$  (the condition to remove  $v_i$ ).  $\square$

Algorithm 2 shows the subroutine **Assign**(**A**, **Y**), which is our DM algorithm. This algorithm

includes three phases: (i) adding all available assignments: find all trips going from RPs to UPs in the current UP set; (ii) assignment trimming: remove all useless trip assignments which map to the trips passing stops contain both RP and UP, and the trips of multiple uploading choices for data from one specific RP; and (iii) reducing lateness: remove all trip assignments that have long travel time to minimize the total penalty of transfer delay according to Eq. (4.3) per Lemma 1.

According to the definition above, we can deduce that the running time of DM algorithm depends on the number of UPCs and trips with time complexity  $\mathcal{O}(N^2V)$ . The running time of UPS algorithm depends on the cost limitation and the number of UPCs and trips. In the worst case, when cost limitation is extremely high, the time complexity of UPS algorithm with subroutine DM is  $\mathcal{O}(N^4V)$ . It is acceptable due to upload points placement and upload path planning are one-time and offline tasks which are time-rich.

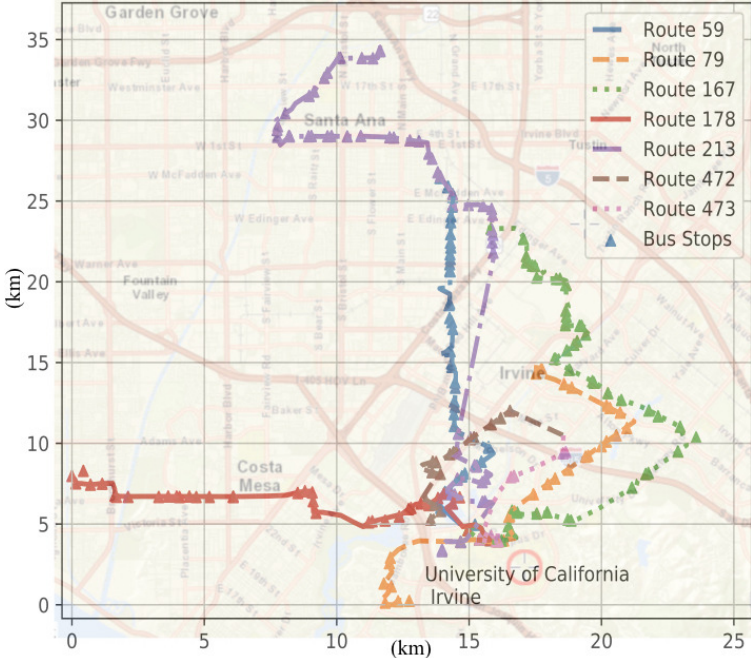


Figure 4.3: Orange country bus routes and stops

## 4.5 Experimental Evaluation of our Approach

We perform simulations to evaluate our proposed algorithms. Our simulation setup consists of four components: (i) data preprocessor, (ii) upload point placement algorithms, (iii) upload path planning algorithms, and (iv) the ONE simulator. The data preprocessor (in Python) converts open transportation datasets into proper formats. We have implemented four upload point placement algorithms: COV, VOL, GA, and UPS algorithms and two upload path planning algorithms, FC and DM algorithms (in Python). All considered algorithms are summarized in Table 4.1. Once the algorithms produce the upload point placement and upload path planning solutions, we put them into Opportunistic Network Environment (ONE) simulator [107]. We modify the ONE simulator to route the sensor data following the solutions from our algorithms and keep track of statistics. We consider the following performance metrics:

- *Penalty value*: The measurement of overall timeliness.
- *Data delivery ratio*: The ratio between the sensor data volumes delivered at UPs and sent by RPs.
- *Late delivery ratio*: The fraction of sensor data that exceeds their delay tolerances.
- *Data transfer time*: The time difference between sensor data arriving at an RP and a UP.
- *Total cost*: The total UP installation cost.
- *Number of UPs*: The number of placed UPs.
- *Running time*: The running time of algorithms.



Table 4.1: Considered Algorithms

	Upload Point Placement			
Upload Path Planning	COV	VOL	GA	UPS
First Contact (FC)	COV <sub>F</sub>	VOL <sub>F</sub>	GA <sub>F</sub>	UPS <sub>F</sub>
Delay Minimization (DM)	COV <sub>D</sub>	VOL <sub>D</sub>	GA <sub>D</sub>	UPS <sub>D</sub>

### 4.5.1 Scenarios

We employ the public transit dataset made public by the Orange County government [141]. The dataset contains bus stop locations, trip schedules, and routes. We focus on the seven bus routes around the UCI campus (shown in Fig 4.3). In particular, our data preprocessor extracts the schedules and bus stop locations for our simulations. The resulting schedule spans a weekday from 6:09 a.m. to 9:09 a.m., which consists of 99 vehicle trips and 551 bus stops in total. The average vehicle trip duration is 64 minutes, while the minimum (maximum) duration is 18 (109) minutes. On average, each vehicle trip traverses through 20.38 stops, and each stop has 35.01 vehicle trips passing by.

We take all the bus stops as our UPCs. The dataset, however, does not contain RPs, nor their data arrival rate, installation cost, and delay tolerance. For each simulation run, we randomly select RPs from all bus stops. We then overlap the dataset with OpenStreetMap to systematically determine the parameters associated with each RP. In particular, for a given RP, we set its data arrival rate to be positively related to the density of surrounding public facilities, which equals  $R_n = len/(\hat{I} \times e^{-fac_n/5})$  where  $fac_n$  is the number of facilities within 300 meters of RP  $n$ ,  $len$  is the length of data packet and  $\hat{I}$  is the maximal data arrival interval. Considering the power supply, we set the installation cost of UPCs depending on their distances to the closest public facilities as  $C_n = \hat{C} \times (1 - e^{-dtf_n/500})$ , where  $dtf_n$  is the distance between UPC  $n$  and its closest public facility and  $\hat{C}$  is the maximal cost. We prioritize the sensing data by setting the delay tolerance to be positively correlated to the

distances from RPs to their closest critical infrastructure (e.g., police station and hospital) following  $D_n = \hat{D} \times (1 - e^{-d_{tc_n}/1000})$  where  $d_{tc_n}$  is the shortest distance from RP  $n$  to critical infrastructures and  $\hat{D}$  is the maximal delay tolerance. We vary the cost limitation between 10 and 640. We consider a small scenario with 20 RPs and a large one with 40 RPs. Simulations with the same inputs and parameters are done with all compared algorithms. Each data point in the figures represents the average of 5 repetitions. In addition, we plot the 1st/3rd quartiles as error bars whenever possible. Table 4.2 lists the detailed simulation parameters.

Table 4.2: Simulation Parameters

Parameter	Value
Simulation time	3 hours
$N$ Number of UPC	551
$V$ Number of trips	99
Number of RP	20 & 40
$\hat{I}$ Maximal data arrival interval	10 s
$\hat{C}$ Maximal installation cost	10
$\Theta$ Cost limitation	10 to 640
$\hat{D}$ Maximal delay tolerance	60 min
$len$ The length of data packet	1 MB
Data transmit rate	1000 MB/s
Data transmit range	20 m
Buffer size of vehicles and RPs	2000 MB

## 4.5.2 Comparison Results

**Our DM algorithm leads to better performance than the FC algorithm with the same upload point placement algorithms.** We compare the two upload path planning algorithms with GA and UPS upload point placement algorithms under cost limitation between 10 and 640. We skip COV and VOL algorithms here because they have the similar results with GA and UPS algorithms. Sample results from 20 RPs are reported; while results

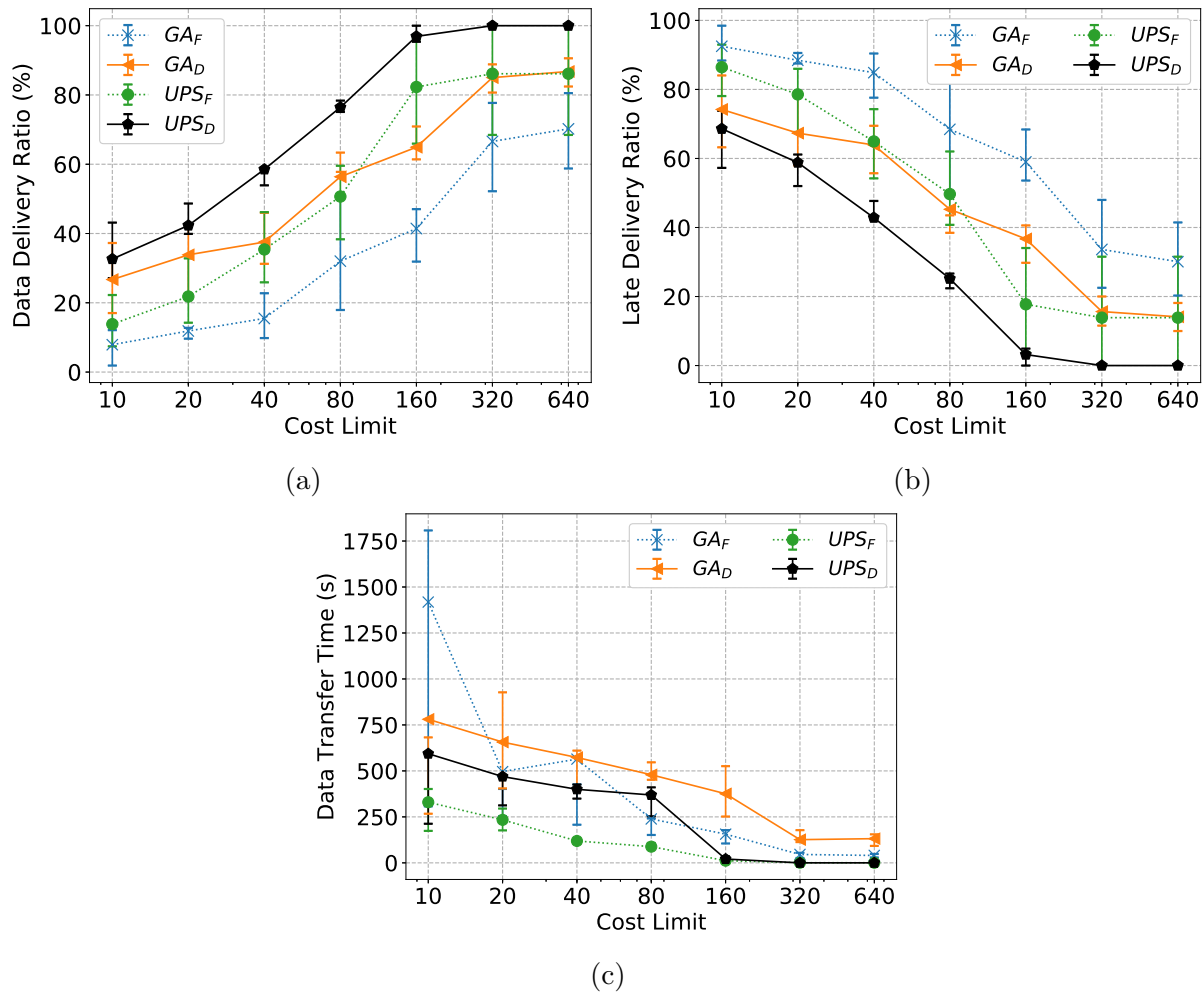
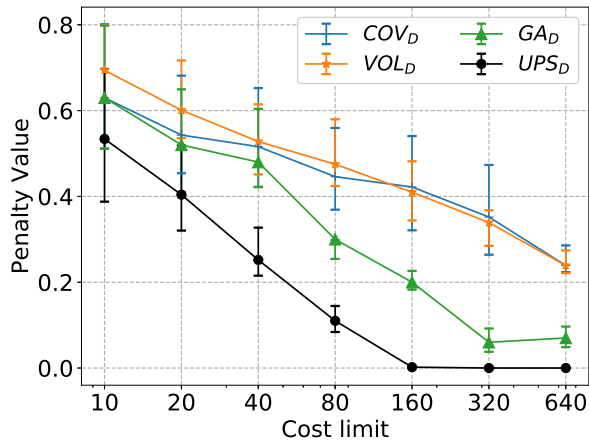


Figure 4.4: Comparisons the performance between the FC and DM algorithms with GA and UPS algorithms under different cost limitations (a) data delivery ratio, (b) late delivery ratio, and (c) data transfer time.

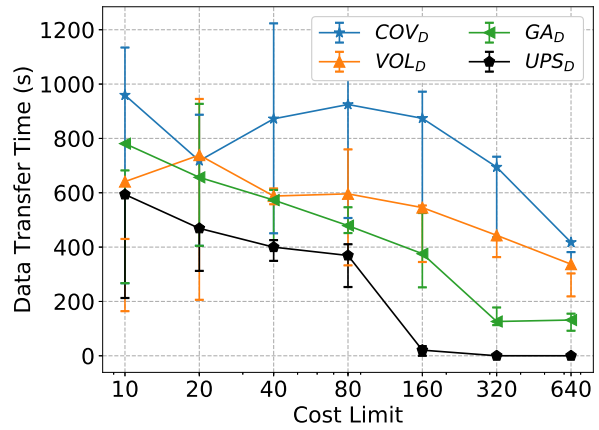
with more RPs are similar. We plot the results in Fig. 4.4. We notice that the penalty value is an output of the upload point placement algorithms, which is common with either upload path planning algorithm. Hence we do not report the penalty value (objective function value) in the figure. We make a few observations on this figure. First, Fig. 4.4a gives the data delivery ratio, which show that our DM algorithm always delivers more data: more than 20% increases are observed. Next, we check if the delivered data are late by looking into the late delivery ratio in Fig. 4.4b. It can be seen that our DM algorithm constantly results in the lower late delivery ratio: 25+% average reduction is possible.

Last, the data transfer time of *delivered data* is given in Fig. 4.4c. This figure depict that the FC algorithm may lead to shorter data transfer time than the DM algorithm. This is because the FC algorithm makes greedy decisions without proper planning, which may occasionally lead to shorter data transfer time. Nonetheless, such difference doesn't change the fact that our DM algorithm delivers: (i) more data and (ii) less late data than the FC algorithm, as shown above. Thus, we no longer consider the FC algorithm in the rest of this work.

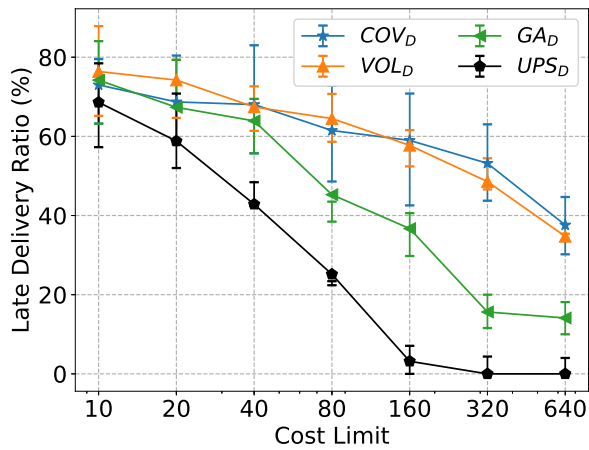
**Our UPS algorithm outperforms other upload point placement algorithms under different cost limitations.** We plot sample results from the four upload point placement algorithms with 20 RPs in Fig. 4.5. In Fig. 4.5a, we observe our proposed UPS algorithm significantly outperforms other algorithms in terms of the objective function value: as high as 20% gap, compared to the GA algorithm is observed. Moreover, as the cost limit increases, UPS algorithm's penalty value descends at a much higher rate than other algorithms, including the GA algorithm. We then check other performance results from the simulators: data transfer time in Fig. 4.5b, late delivery ratio in Fig. 4.5c, and data delivery ratio in Fig. 4.5d. In all these figures, our UPS algorithm outperforms other algorithms, and the performance gap becomes nontrivial even with a moderate cost limitation. For example, with a cost limit of 160, compared to other algorithms, our UPS algorithm achieves sub-21 sec data transfer time (15+ times improvement), sub 3.2% late delivery ratio (about 12 times improvement),



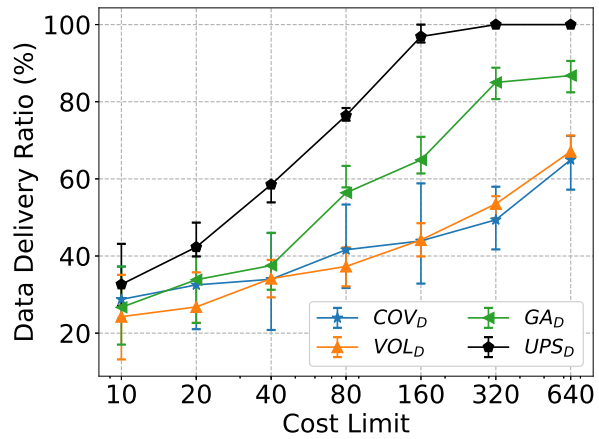
(a)



(b)



(c)



(d)

Figure 4.5: Performance of the four upload point placement algorithms under different cost limitations: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio.

and above 96% data delivery ratio (about 50% improvement).

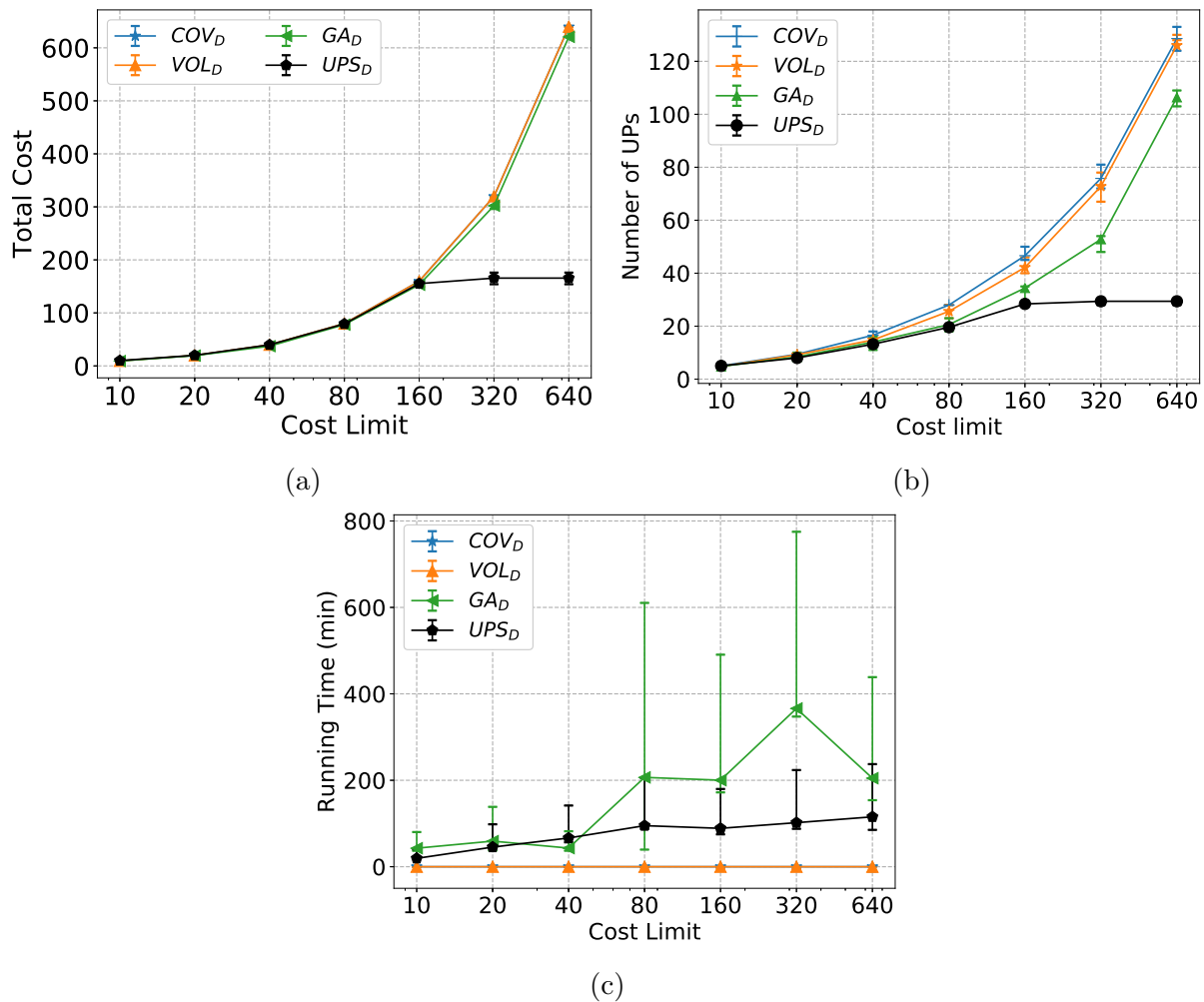


Figure 4.6: Installation cost and running time of the four upload point placement algorithms under different cost limitations: (a) total cost, (b) number of UPs and (c) running time.

**Our UPS algorithm results in cost-effective upload point placement.** We observe above that our UPS algorithm achieves better performance with a rather small increase in cost limitation. We next dig a bit deeper and plot the total cost of the four algorithms from 20 RPs in Fig. 4.6a. This figure shows that our UPS algorithm only consumes a total cost of about 180, even when the cost limitation is beyond that. Fig. 4.6b also demonstrates that the UP placement decisions are almost frozen beyond the cost limitation of 160. These two figures demonstrate that our UPS algorithm makes cost-effective placement decisions;

on top of its superior performance. In contrast, three other algorithms continue to use up all the cost limitation yet deliver inferior performance. **Our UPS algorithm has relatively**

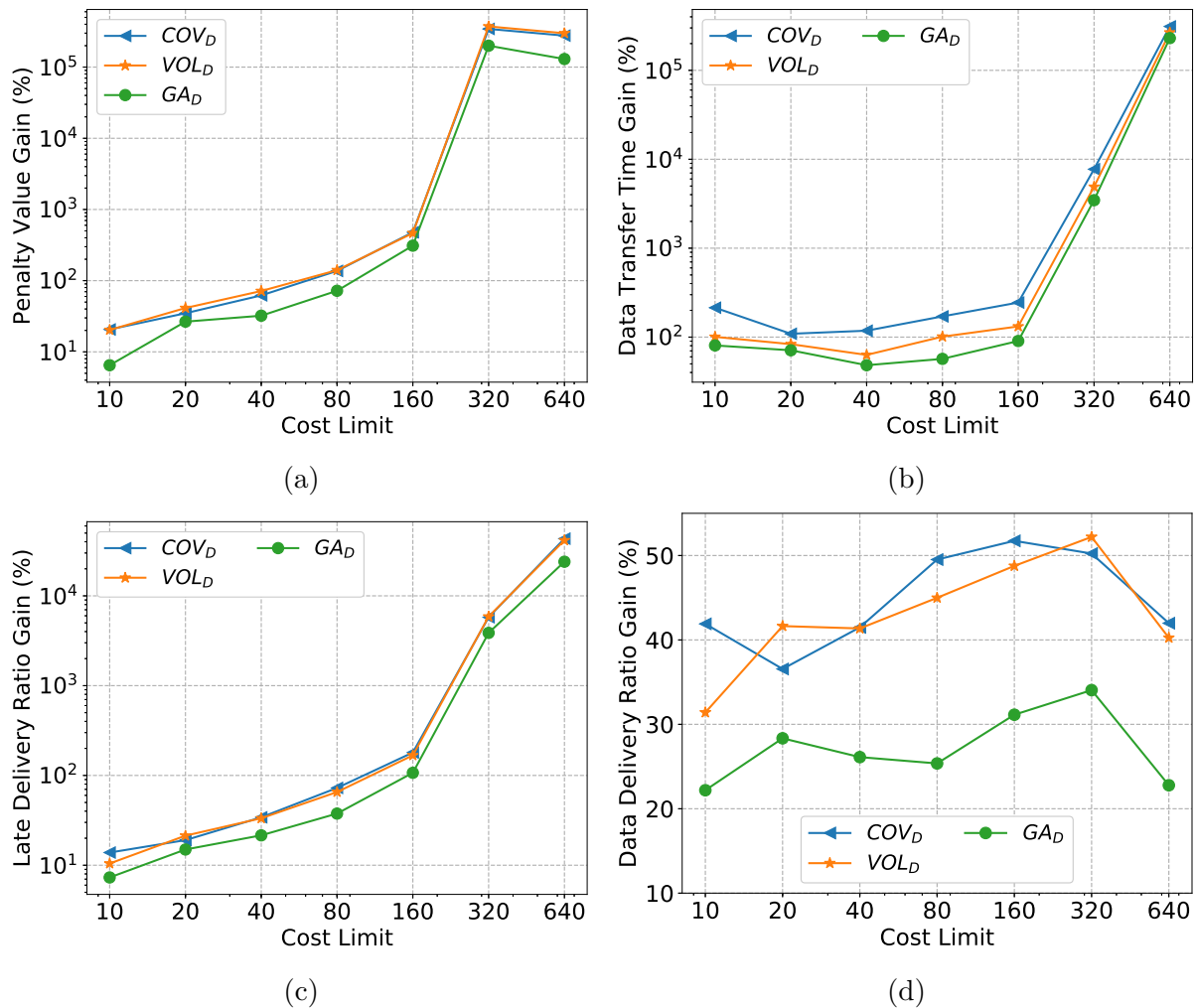


Figure 4.7: The performance gains of UPS comparing with the other algorithms on: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio under different cost limitations in the scenario with 40 RPs.

**shorter running time compared with GA and better performance** We next compare the running time of four upload point placement algorithms with 20 RPs in Fig. 4.6c. It is shown that the convergence time of GA is about twice the running time of our UPS algorithm when cost limitation greater than 40, which fluctuates between 200 to 600 minutes while the running time of UPS is steady at about 100 minutes.

**Our UPS algorithm delivers prominent performance gains over other algorithms**

**in larger/heavier scenarios.** We next report performance *gain* of our UPS algorithm over other algorithms, which is defined as the performance improvement normalized to UPS' value. Notice that, our UPS algorithm may achieve zero late delivery ratio. In that case, we put 0.01% in the denominator to be conservative. Fig. 4.7 shows the sample performance gains from the larger scenario. This figure confirms our above observations on the smaller scenario are also applicable to larger scenario. Even through with more RPs (i.e., heavier traffic), the performance gains remain significant across the considered cost limitations. For example, at the cost limitation of 160, our UPS algorithm gets at least 478% gain in penalty value, 100% gain in data transfer time, 100% gain in late delivery ratio, and 32% gain in data delivery ratio. **Hence, we recommend the combination of UPS and DM algorithms for solving the upload point placement and upload path planning problems.**

## 4.6 Summary and Discussion

In this chapter, we studied the use of scheduled transportation fleets to enable cost-effective, reliable, and timely data collection in urban IoT settings with limited backhaul connectivity. We illustrated the value of a hierarchical approach that includes (a) the creation of locally connected “IoT zones.” with planned collection points (RPs), (b) careful positioning of limited upload points from which data is uploaded to backend data processing centers, and (c) intelligent planning of data movement from RPs to UPs using already scheduled transportation fleets. In the upcoming chapter, our aim is to expand our focus to encompass urban scenarios that include extreme conditions like fires and earthquakes. These events hold the potential to disrupt the sensing, communication, and transportation infrastructure within smart communities. Moreover, we will delve into the feasibility of implementing alternative data transfer methods, including drones. We aim to explore the feasibility of deploying alternative data transfer methods, such as drones, to facilitate aerial data collection and



transmission. This approach aims to enhance real-time data delivery and situational awareness during extreme conditions. Our goal is to design and implement IoT platforms that support drone-based data collection processes. These platforms will also offer user interfaces for controlling drone movements, providing a comprehensive solution for effective data gathering and analysis during challenging urban scenarios.

## Chapter 5

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

This chapter explores the utilization of on-demand drone mobility to enhance the sensing capabilities of IoT applications. We introduce DragonFly, a drone-based mobile sensing framework specifically designed to improve real-time situational awareness in high-rise buildings, with a primary focus on monitoring 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).

## 5.1 Chapter Overview

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 [69] - 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 chapter, 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 [143]. 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 firefighting 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 [188]. 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 [154]. 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 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 [155]. Particularly, sudden loss of windows or similar building envelope components are known to create unsurvivable situations [65]. Significant research confirmed the intensity of these fires due to window loss: literally, changing the temperature by thousands of degrees (F) in tens of seconds from floor to ceiling. Such conditions greatly exceed the limit of today’s firefighter PPE (Personal Protective Equipment), leading to internal conditions that are not survivable to firefighters. 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-rise fire, where the fires may rapidly spread along the exterior surface [222]. There have been multiple such high-rise building fires, e.g., the 2017 Grenfell Tower fire in London with 72 fatalities [147].

In this work, 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 [156]. 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 [218]. 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 work makes the following contributions:

- DragonFly system design for high-rise fire monitoring (Sec. 5.3).
- 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. 5.4).
- Development of a two-step approach to solve the MWSP (Sec. 5.5).
- Evaluations of our solution using simulations (Sec. 5.6).

## 5.2 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 cameras, 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.** We supplement the related work discussed in Section 2.4 by providing a more detailed technical discussion concerning the use of drones to improve mobile sensing. Drones have frequently been utilized for aerial sensing and surveillance during both non-disaster and disaster times – e.g., building surveillance [177] and post-disaster environment assessment [64]. Drones carrying specific payloads (e.g., fire retardants and extinguishing balls) have been used in wildland fire scenarios [91], 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 [166]. 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 [47]. 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 [220] and patrolling multiple regions with changing features at different rates [200]. Region partitioning techniques in conjunction with inter-region waypoint scheduling [194] 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 [219]. A range of heuris-

tic approaches using Mixed Integer Linear Programming (MILP), Markov Decision Process (MDP), and game theory have illustrated the complexity of the problem [142]. 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 [23] and hostile environments [114]. For example, the techniques for drones to rendezvous at unspecified locations [134] or capture geo-dispersed targets in no-fly zones [127] 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 work introduces new levels of complexity as follows. First, the environment is dynamic due to the fire spread, human movements, and changes in the 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 to detect fire spread.

Finally, the locations of drones and their distances from the buildings affect the monitoring performance and detecting coverage. Recent studies [122, 45, 199] 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 from 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, which leads to a smaller image coverage but more accurate event detection.

Because of the above concerns, we formulate 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 [220, 200, 194], 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.

### 5.3 The DragonFly Framework

We next provide an overview of the DragonFly framework, which is shown in Fig. 5.1.

**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. Sample events include the presence/absence of fire, the existence of humans and the open



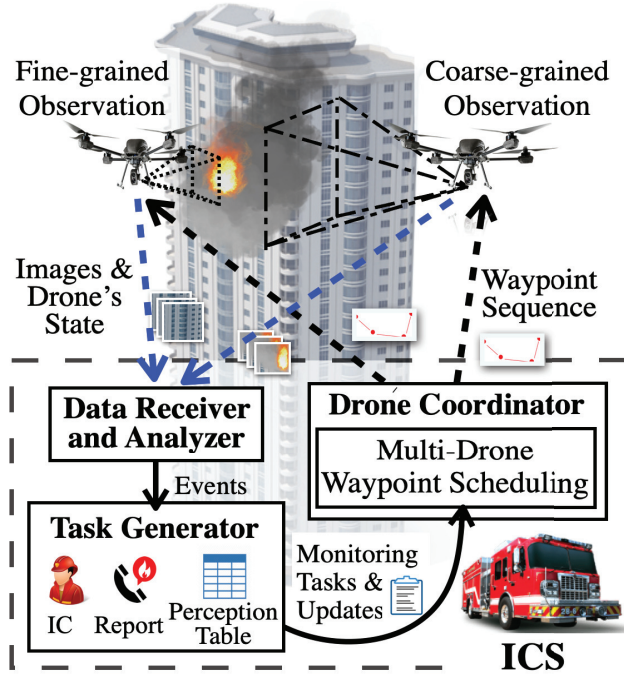


Figure 5.1: Overview of DragonFly framework.

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 5.1 for an example) and automatically updated to record new detections (see Table 5.2 for an example).

**Task generator.** *Task generator* creates monitoring tasks stored in a task table (see Table 5.3 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 fire-fighting 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 update 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 [218]. Table 5.3 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.

Table 5.1: Initial Perception

Event	Location	Time
Fire	R601	10:00

Table 5.2: 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 5.3: 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

**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 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 thesis) 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 work, we assume fire fighting forces have access to the high-rise building structure information, such as the floorplans, locations of windows, and mapping between windows and rooms. We believe such assumptions are not too strong, as firefighting forces are part of local governments, while open-data paradigm [157] 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 [234]. This is also reasonable given the rapid increase in the penetration rate of mobile networks and reliable drone APIs such as DJI Mobile SDK [61]. Last, we assume the design of image analysis algorithms are orthogonal to this work, any image (or sensor) data analysis algorithms can be adopted by DragonFly, and their accuracy levels are given.

## 5.4 Multi-Drone Coordination for High-Rise Fires

This works focus on the implementation of the drone coordinator to tackle the multi-drone waypoint scheduling problem (MWSP). The MWSP entails planning the motions of multiple drones to complete assigned tasks generated by a task generator within a specified plan duration. To address this challenge, we define essential notations for mathematical representation and proceed to formulate this problem into an NP-hard optimization problem.

### 5.4.1 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  $\sigma_k$ , and desired frequency  $\eta_k$ . 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  $\mathbf{M} = \{m_1, \dots, m_M\}$  indicate the set of all possible monitoring areas at our fire scene. Each monitoring area is defined as  $m_i = (\{v_1^i, v_2^i, v_3^i, v_4^i\}, \vec{n}_i)$ ,  $i = 1, \dots, M$ , 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.

### 5.4.2 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  $\mathbf{W} = \{w_1, \dots, w_W\}$ , 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<sup>1</sup> 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 [35] with elevation difference.

### 5.4.3 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  $d - A(d, e_k)$ , 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

---

<sup>1</sup>Denser waypoints can be considered at the cost of higher computational complexity.

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  $\mathbf{T}^{ar} = [t_1, t_2, \dots]$  at waypoints  $[w(t_1), w(t_2), \dots]$  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:

$$OA_k(t) = \begin{cases} A(d_{w(t)}, e_k) \times C(w(t), o_k), & t \in T^{ar} \\ 0, & \textit{otherwise.} \end{cases} \quad (5.1)$$

From Eq. (5.1), 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  $\eta_k$ , 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

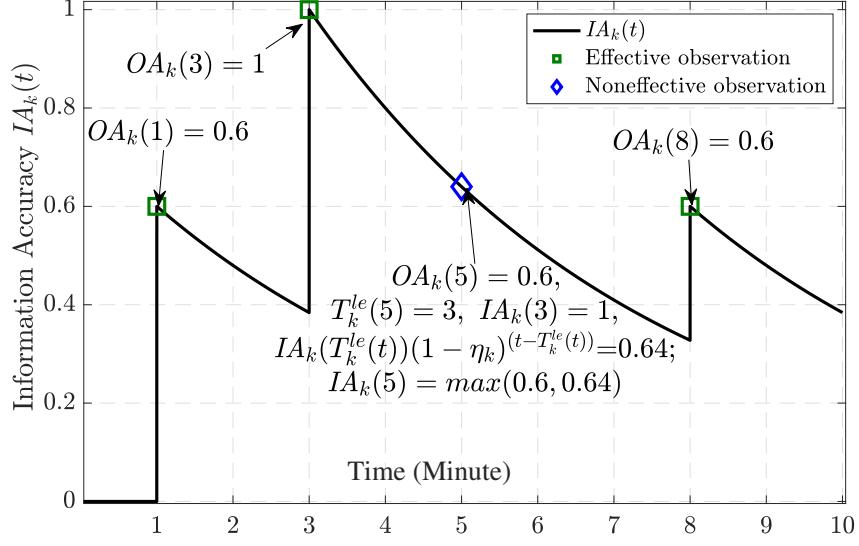


Figure 5.2: Sample  $IA_k(t)$  with  $\eta_k = 0.2$  and  $g_k = 0$ .

Fig. 5.2:

$$IA_k(t) = \begin{cases} 0, & t = g_k; \\ \max\{IA_k(T_k^{le}(t))(1 - \eta_k)^{(t-T_k^{le}(t))}, OA_k(t)\}, & t > g_k. \end{cases} \quad (5.2)$$

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  $[g_k, t']$ .

#### 5.4.4 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  $[w_{in(1)}, \dots, w_{in(N)}]$  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  $\mathbf{T}_n^{ar}(\mathbf{X}) = [T_n^{ar}(1), \dots, T_n^{ar}(S)]$ , 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:

$$T_n^{ar}(s) = t_0 + \sum_{s'=1}^s \sum_{w_i, w_j \in \mathbf{W}} \left( \frac{d(\langle w_i, w_j \rangle)}{R_{fly}} + T_{loi} \right) x_{i,j}^{n,s'}, \quad (5.3)$$

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:

$$OA_k^n(T_n^{ar}(s), \mathbf{X}) = \sum_{w_i, w_j \in \mathbf{W}} C(w_j, o_k) A(d_{w_j, e_k}) x_{i,j}^{n,s}. \quad (5.4)$$

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. (5.1) by considering the possibility that multiple drones cover a area simultaneously. The new observation accuracy is:

$$OA_k(t, \mathbf{X}) = \max_{n \in [1, N]} \{OA_k^n(t, \mathbf{X})\} \quad (5.5)$$

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:

$$IA_k(t, \mathbf{X}) = \begin{cases} IA_k(T_k^{le}(t_0))(1 - \eta_k)^{(t_0 - T_k^{le}(t_0))}, & t = t_0; \\ \max_{n \in [1, N]} \{IA_k(T_k^{le}(t))(1 - \eta_k)^{(t - T_k^{le}(t))}, OA_k^n(t, \mathbf{X})\}, & t > t_0. \end{cases} \quad (5.6)$$

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:

$$\int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{X})dt = \int_{t=g_k}^{T_k^{le}(t_0)} IA_k(t)dt + \int_{t=T_k^{le}(t_0)}^{t_0+T} IA_k(t, \mathbf{X})dt. \quad (5.7)$$

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:

$$\max \min_{k \in \mathbf{K}} \left\{ \frac{1}{\sigma_k} \int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{X}) dt \frac{1}{t_0 + T - g_k} \right\} \quad (5.8a)$$

$$\text{s.t. } \sum_{s=1}^S \left( \frac{d(\langle w_i, w_j \rangle)}{R_{fly}} + T_{loi} \right) \sum_{w_i \in \mathbf{W}} \sum_{w_j \in \mathbf{W}} x_{i,j}^{n,s} \leq T; \quad (5.8b)$$

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

$$\forall w_h \in \mathbf{W} \setminus \{w_0\}, T_n^{ar}(s) = T_{n'}^{ar}(s'), n \neq n'; \quad (5.8c)$$

$$\sum_{w_i \in \mathbf{W}} \sum_{w_j \in \mathbf{W}} x_{i,j}^{n,s} = 1; \quad (5.8d)$$

$$\sum_{w_j \in \mathbf{W}} x_{in(n),j}^{n,1} = \sum_{w_i \in \mathbf{W}} x_{i,0}^{n,S} = 1; \quad (5.8e)$$

$$\sum_{w_i \in \mathbf{W}} x_{i,0}^{n,s} \leq x_{0,0}^{n,s+1}; \quad (5.8f)$$

$$\sum_{w_i \in \mathbf{W}} x_{i,j}^{n,s''} = \sum_{w_z \in \mathbf{W}} x_{j,z}^{n,s''+1}, \forall s'' \in [1, S-1]; \quad (5.8g)$$

$$x_{i,j}^{n,s} \in \{0, 1\}; \quad (5.8h)$$

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

The objective function in Eq. (5.8a) 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. (5.8b) ensures that the total time spent by each drone is within the plan duration  $T$ . Eq. (5.8c) 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. (5.8d). The constraints in Eqs. (5.8e) and (5.8f) set the initial and final waypoints for all drones. Eq. (5.8g) ensures the connectivity of the generated waypoint sequences. Last, Eq. (5.8h) 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) [75] 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. (5.8a) 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

$$\max \sum_{k \in \mathbf{K}} \int_{t=g_k}^{t_0+T} \sigma_k IA_k(t, \mathbf{X}) dt \frac{1}{t_0 + T - g_k}, \quad (5.9)$$

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 work because the worst-case scenario carries much higher weight in high-rise fires.

## 5.5 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. 5.3 illustrates the workflow of our MWSP solution, which is detailed in the following.

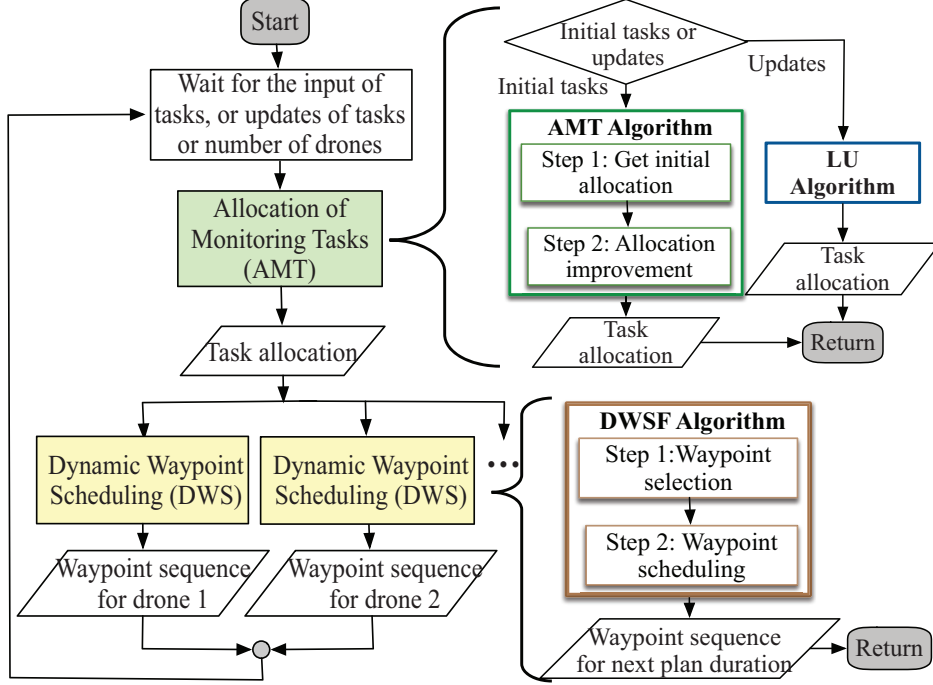


Figure 5.3: Workflow of our proposed algorithms.

### 5.5.1 Allocation of Monitoring Tasks: AMT

Given monitoring tasks  $\mathbf{K}$ , we first derive the set of monitoring areas:  $\mathbf{M}^{\mathbf{K}}$  where  $\mathbf{M}^{\mathbf{K}} \subseteq \mathbf{M}$  and  $o_k \in \mathbf{M}^{\mathbf{K}} \forall k \in \mathbf{K}$ . The AMT problem spatially allocates  $\mathbf{M}^{\mathbf{K}}$  into  $N$  disjoint subsets  $\mathbf{M}' = \{\mathbf{M}'_1, \dots, \mathbf{M}'_N\}$ , with  $\mathbf{M}^{\mathbf{K}} = \bigcup_{n=1}^N \mathbf{M}'_n$ . Given  $\mathbf{M}'$ , we allocate all monitoring tasks  $\mathbf{K}$  into  $N$  disjoint subsets represented by  $\mathbf{K}' = \{\mathbf{K}'_1, \dots, \mathbf{K}'_N\}$  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  $\mathbf{G} = (\mathbf{M}^{\mathbf{K}}, \mathbf{E})$ , where the nodes indicate the monitoring areas  $\mathbf{M}^{\mathbf{K}}$ , and edges  $\mathbf{E} = \{\langle m_i, m_j \rangle \mid m_i, m_j \in \mathbf{M}^{\mathbf{K}}\}$  are the pairwise links between any two monitoring areas. We use the center of area  $m \in \mathbf{M}^{\mathbf{K}}$  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<sup>2</sup>.

Based on MWSP's objective function in Eq. (5.8a), 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  $m_i \in \mathbf{M}^{\mathbf{K}}$  throughout plan duration  $T$  as  $B(m_i)$ , which is a function of task significance and frequency. More precisely, we write it as:

$$B(m_i) = \begin{cases} 1, & \max_{k \in \mathbf{K}, o_k = m_i} \{\eta_k\} = 0, \exists k \in \mathbf{K} : o_k = m_i; \\ \lceil T \times \max_{k \in \mathbf{K}, o_k = m_i} \{\sigma_k \times \eta_k\} \rceil, & \textit{otherwise}. \end{cases} \quad (5.10)$$

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:

$$ET(\mathbf{M}'_n) = \sum_{m_i \in \mathbf{M}'_n} B(m_i) \left( \frac{2 \times d(\langle m_i, \hat{m}_n \rangle)}{R_{fly}} + T_{loi} \right), \quad (5.11)$$

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

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

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

$$\min \max_{\mathbf{M}'_n \in \mathbf{M}'} \{ET(\mathbf{M}'_n)\}. \quad (5.12)$$

---

<sup>2</sup>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. 5.4.1.

The AMT problem is also NP-hard, which can be proven by reducing a load balancing problem [116] 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 [193] 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 [164] 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 3.

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.

---

**Algorithm 3:** 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  $\mathbf{M}' = \{\mathbf{M}'_1, \dots, \mathbf{M}'_N\}$ , and task allocation:  
 $\mathbf{K}' = \{\mathbf{K}'_1, \dots, \mathbf{K}'_N\}$ .

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

- 1 Greedily select the initial medoids  $\{\hat{m}_1, \dots, \hat{m}_N\}$ .
- 2 Get the initial clustering:  $\mathbf{M}' = \{\mathbf{M}'_1, \dots, \mathbf{M}'_N\}$  using Voronoi Iteration [164].
- 3 Get  $B(m_i)$  for all  $m_i \in \mathbf{M}^{\mathbf{K}}$  by Eq. (5.10).
- 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                  $BestNeigh \leftarrow \mathbf{M}'_{new}$ ;  $Min \leftarrow Min'$ ;  $Sum \leftarrow Sum'$ .
- 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  $\mathbf{K}'$ .

---

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 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  $\mathcal{O}(N_a|\mathbf{M}^{\mathbf{K}}|^2)$ .

### 5.5.2 Dynamic Waypoint Scheduling: DWS

Upon getting task allocation  $\mathbf{K}' = \{\mathbf{K}'_1, \dots, \mathbf{K}'_N\}$  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:

$$\max \min_{k \in \mathbf{K}'_n} \left\{ \frac{1}{\sigma_k} \int_{t=g_k}^{t_0+T} IA_k(t, \mathbf{Y}) dt \frac{1}{t_0 + T - g_k} \right\}. \quad (5.13)$$

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 [40] 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. (5.6). 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 4.

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.



---

**Algorithm 4:** 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  $[t_0, t_0 + T]$ .

```

1  $\mathbf{P} \leftarrow [w_{in(n)}]; t \leftarrow t_0, w' \leftarrow w_{in(n)}, \mathbf{W}' = \emptyset$ 
  /* Step 1: Select waypoints to cover all areas. */
2 for  $d \in \mathbf{D}$  do
3    $\mathbf{M}''_n \leftarrow \mathbf{M}'_n; \mathbf{W}_d \leftarrow \{w_i | w_i \in \mathbf{W}, d_{w_i} = d\}$ .
4   while  $\mathbf{M}''_n \neq \emptyset$  do
5     Get  $w_i = \arg \max_{w_a \in \mathbf{W}_d} \{|\{m | m \in \mathbf{M}''_n, C(w_a, m) = 1\}|\}$ .
6      $\mathbf{W}' \text{.add}(w_i); \mathbf{M}''_n \leftarrow \mathbf{M}''_n \setminus \{m | m \in \mathbf{M}''_n, C(w_i, m) = 1\}$ .
  /* Step 2: Waypoint scheduling. */
7 while  $t + d(\langle w', w_0 \rangle) / R_{fly} < t_0 + T$  do
8    $Min \leftarrow \min_{k \in \mathbf{K}'_n} \{\frac{1}{\sigma_k} AUC(k)\}$ .
9    $\hat{\mathbf{M}} \leftarrow \{o_k | k \in \mathbf{K}'_n, \frac{AUC(k)}{\sigma_k} = Min\}; Count \leftarrow |\hat{\mathbf{M}}|$ .
  /* Predict the AUC if drone visits a waypoint. */
10 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
11   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. (5.6).
12    $Min[i] \leftarrow \min_{k \in \mathbf{K}'_n} \{\frac{1}{\sigma_k} AUC'(k)\}$ .
13    $\hat{\mathbf{M}}[i] \leftarrow \{o_k | k \in \mathbf{K}'_n, AUC'(k) / \sigma_k = Min[i]\}$ .
14    $Count[i] \leftarrow |\hat{\mathbf{M}}[i]|$ .
  /* Select the next waypoint. */
15 if  $\min_{w_i \in \mathbf{W}} \{Min[i]\} < Min$  then
16    $\rho \leftarrow \arg \max_{w_i \in \mathbf{W}'} \frac{Min - Min[i]}{d(\langle w', w_i \rangle) / R_{fly} + T_{loi}}$ .
17 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. */
18  $t \leftarrow t + d(\langle w', \rho \rangle) / R_{fly} + T_{loi}; w' \leftarrow \rho; \mathbf{P} \text{.add}(\rho)$ .
19 for  $k \in \{k' | k' \in \mathbf{K}'_n, C(\rho, o_{k'}) = 1\}$  do
20   Update  $l_k(t), R_k(l_k(t))$  and  $AUC(k)$  by Eq. (5.6).
21 return  $\mathbf{P}$ 

```

---

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  $\mathcal{O}(N_c |\mathbf{W}|^2 + T |\mathbf{W}'| |\mathbf{K}|)$ .

## 5.6 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.

### 5.6.1 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 [193] 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 [70], and (iii) the Minimum Spanning Tree (MST) algorithm, which also generates a recurring TSP tour using a minimum spanning

tree approximation [85]. We consider all pairs of the allocation and scheduling algorithms, as illustrated in Table 4.1.

Table 5.4: Considered Algorithmic Combinations

Scheduling Allocation	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 [122, 45]. Table 5.5 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. 5.4. 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 [46]. 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.

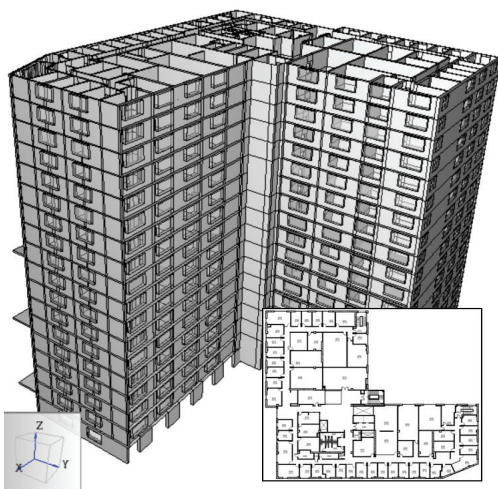


Figure 5.4: The building used in our simulations.

Table 5.5: 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

Table 5.6: Task Types

Type	Event ( $e_k$ )	Sig. ( $\sigma_k$ )	Fre. ( $\eta_k$ )
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 5.7: 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

According to the practical monitoring requirements [218], 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 and tracks the human movements. We set following rules to generate monitoring tasks, based on the task types in Table 5.6.

- (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  $\Theta_a$ . 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 5.7 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.

## 5.6.2 Simulation Results

**Scheduling algorithms.** Figs. 5.5 and 5.6 compare our proposed DWSF with other baseline algorithms when using the AMT algorithm for task allocation. We give sample results

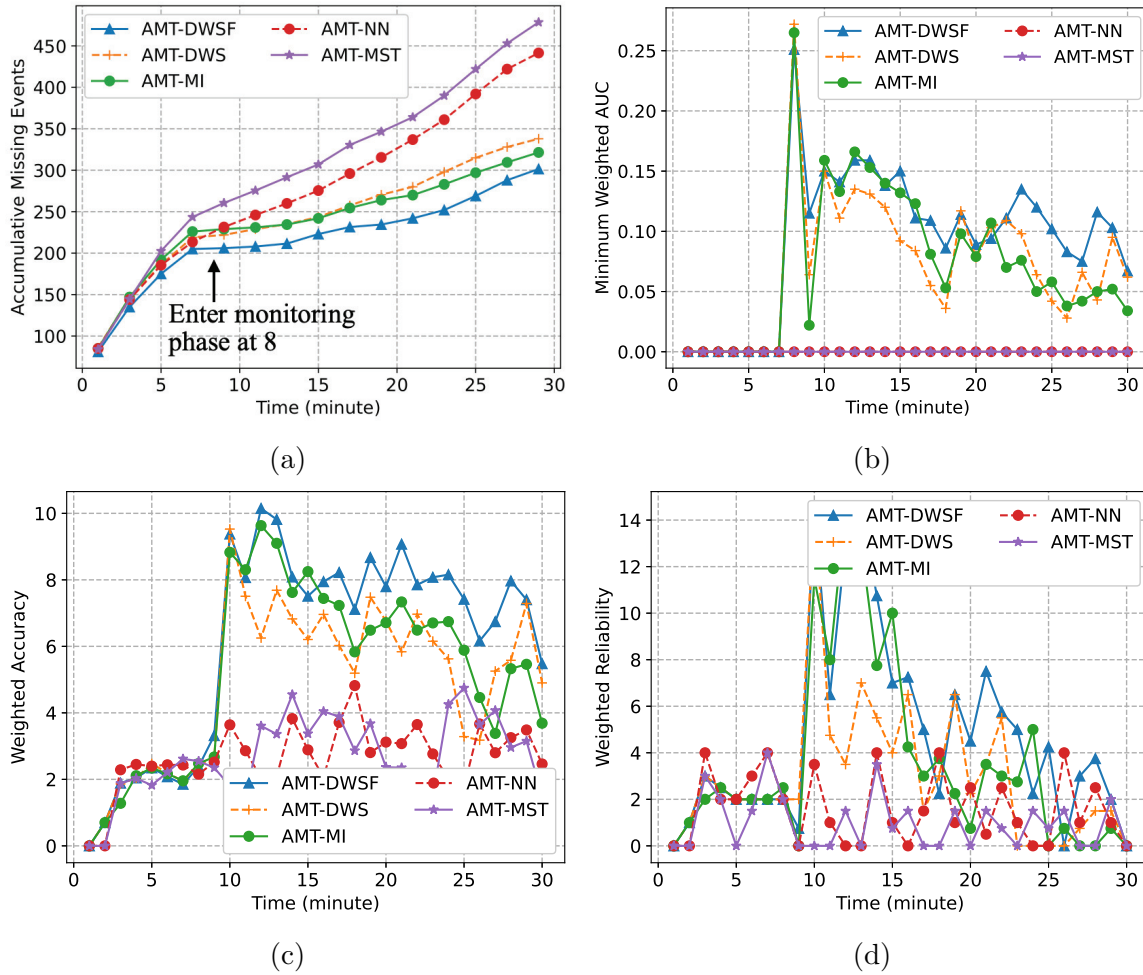
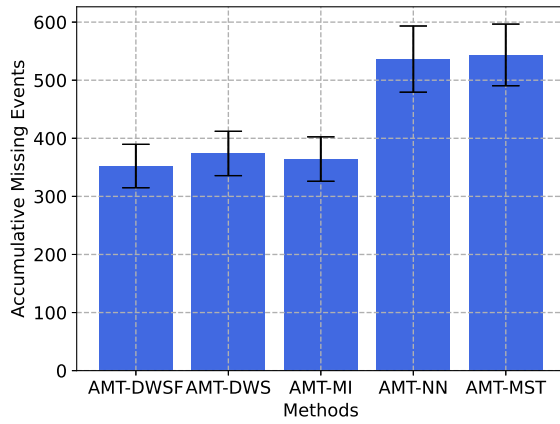
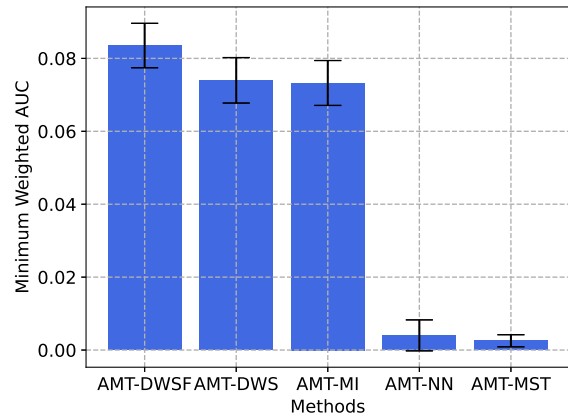


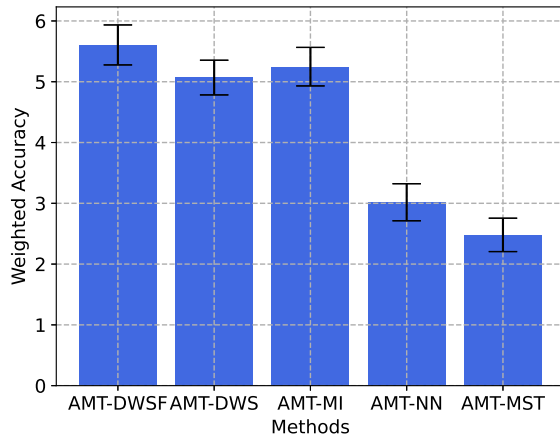
Figure 5.5: Performance of DragonFly throughout a sample simulation on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability.



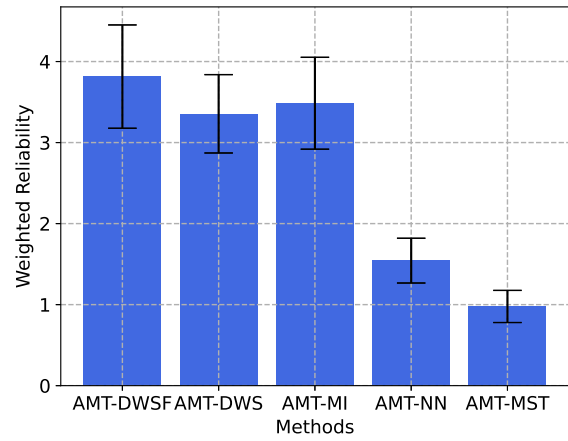
(a)



(b)



(c)



(d)

Figure 5.6: Performance of DragonFly across 25 runs on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability.

from simulations with 5 drones and 177 tasks. Figs 5.5 reveal that our AMT-DWSF algorithm always outperforms the baseline scheduling algorithms in all aspects throughout a sample simulation run. Figs 5.6 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. 5.5 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. 5.5(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. 5.5(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. 5.7 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. 5.7(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. 5.7 reveals the superior performance of our proposed AMT algorithms, compared to the baseline ones.*



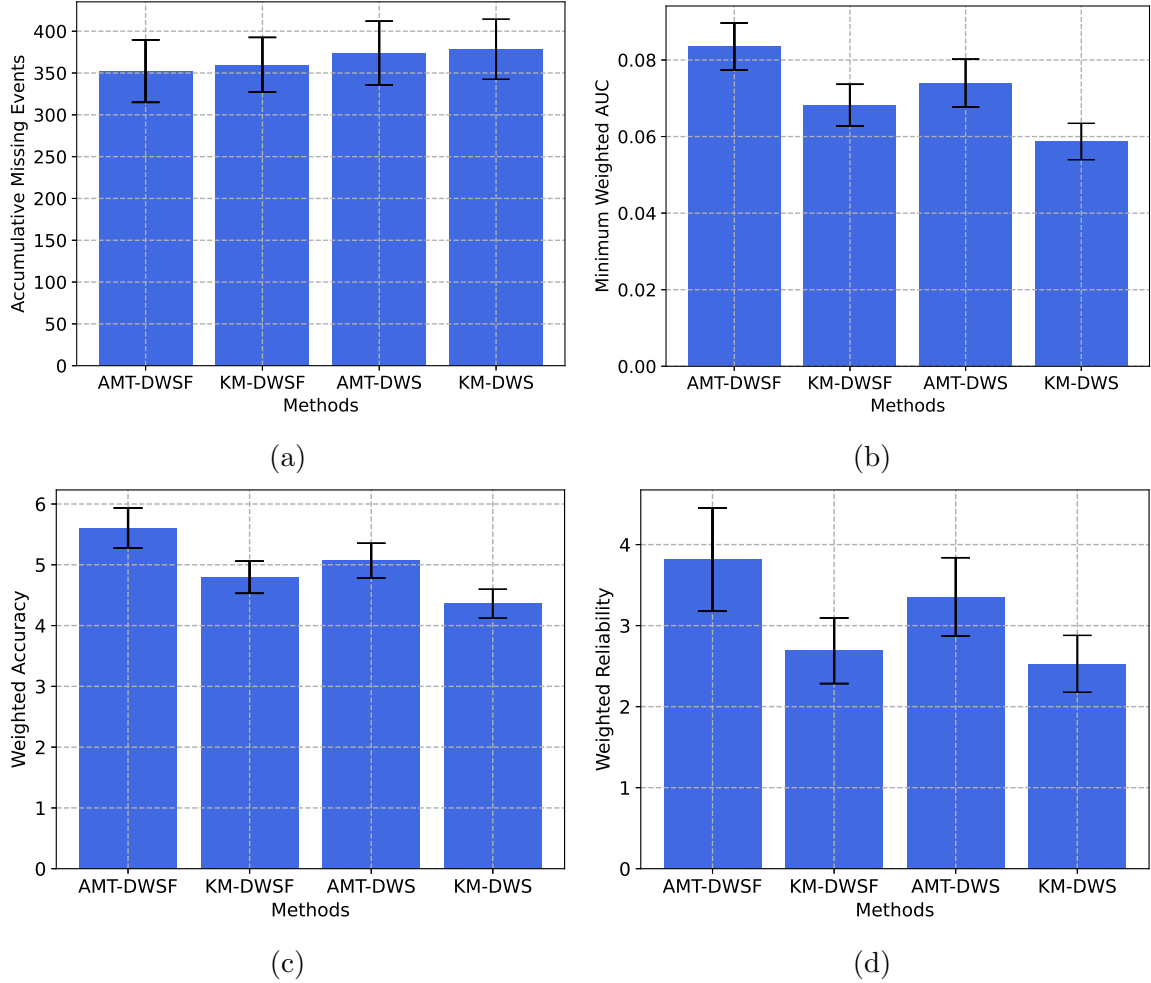
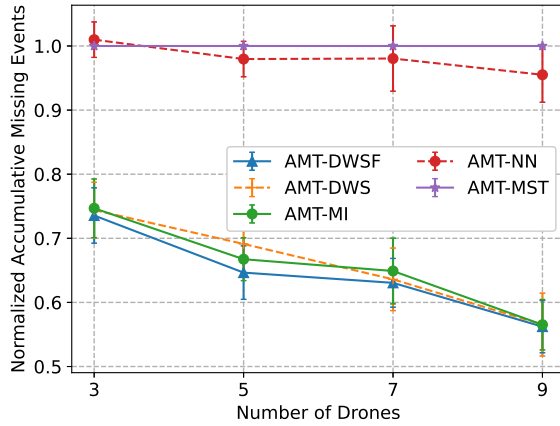
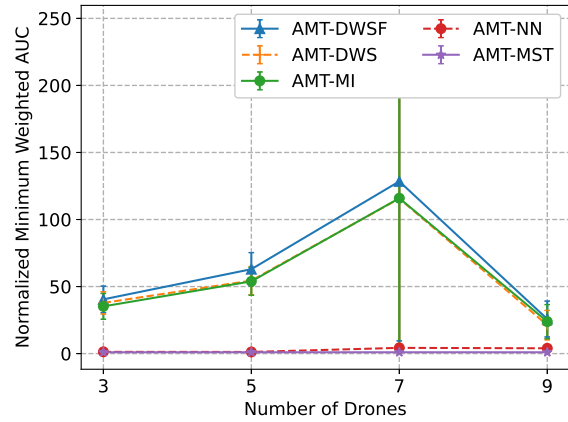


Figure 5.7: Impact of task allocation strategy on: (a) accumulative missing events, (b) minimum weighted AUC, (c) weighted accuracy, and (d) weighted reliability.

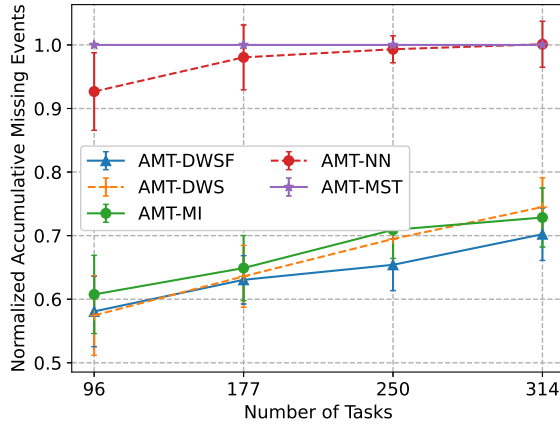
**Scalability of our proposed algorithm.** We next consider larger MWSP problems with different number of drones (between 3 and 9) and different number of tasks (between 96 to 314). First, we employ the AMT algorithm for the allocation problem and compare the performance of scheduling algorithms in Fig. 5.8. Figs. 5.8(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 AUC (by up to 110 times), compared to the AMT-MST algorithm. Figs. 5.8(c) and (d) depict the statistics of the various scheduling algorithms under different number of tasks. Our AMT-DWSF algorithm



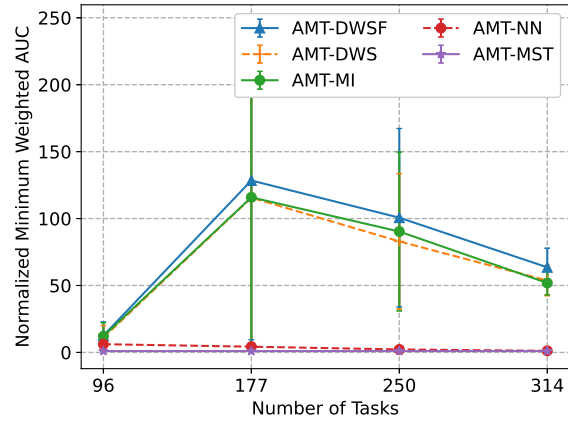
(a)



(b)



(c)



(d)

Figure 5.8: 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.

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. 5.8, 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. 5.9. Figs. 5.9(a) and 5.9(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. 5.9(c) and 5.9(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 the KM-DWSF, achieves higher weighted accuracy (about 1.2 times), and higher weighted reliability (about 1.9 times) than KM-DWSF. *Fig. 5.9 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. 5.10 reports the running time of the LU algorithm plus the waypoint scheduling algorithms under diverse scenarios. The running times are

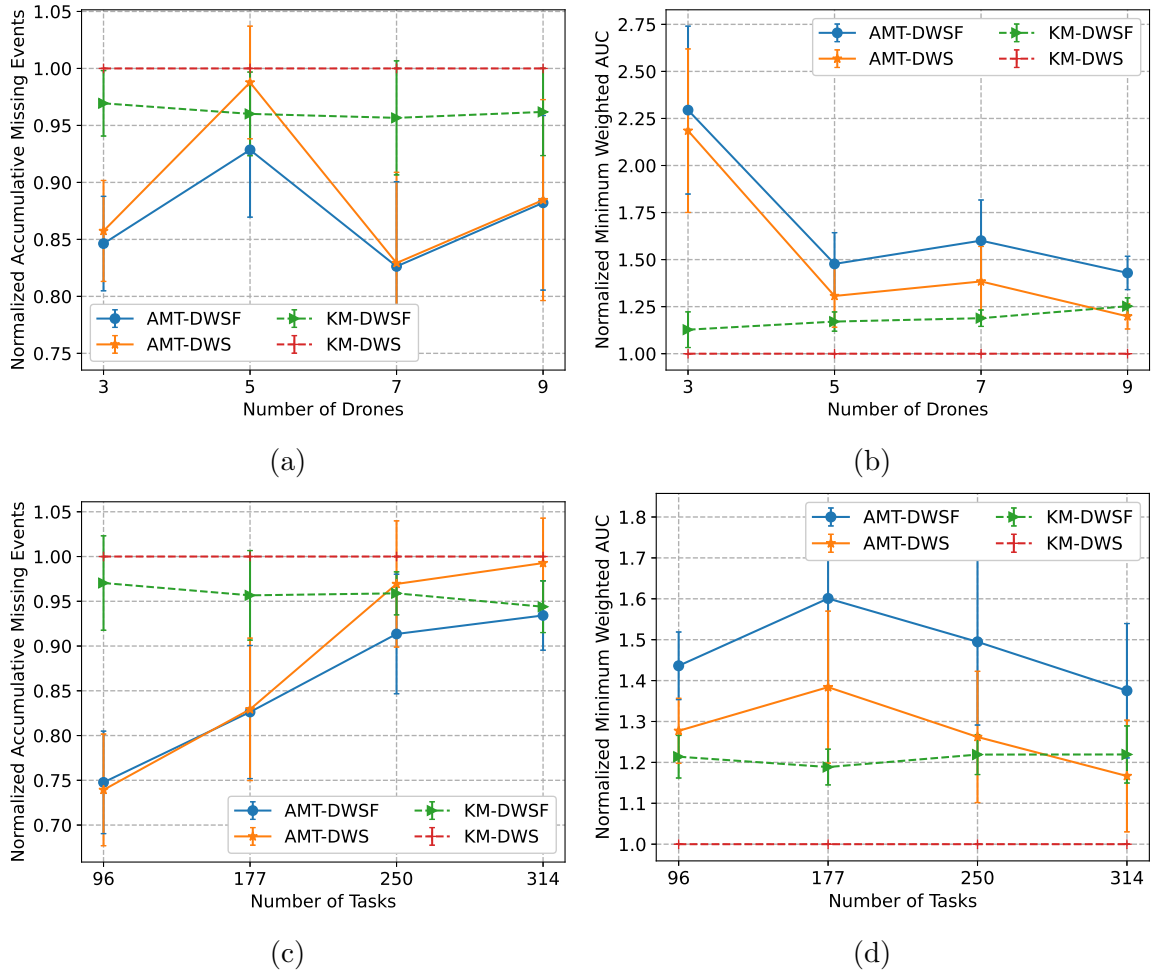


Figure 5.9: Impact of task allocation strategy under diverse problem sizes: (a), (b) 177 tasks with 3 to 9 drones, (c), (d) 7 drones with 96 to 324 tasks. (a), (c) give normalized accumulated missing events, and (b), (d) give normalized minimum weighted AUC.

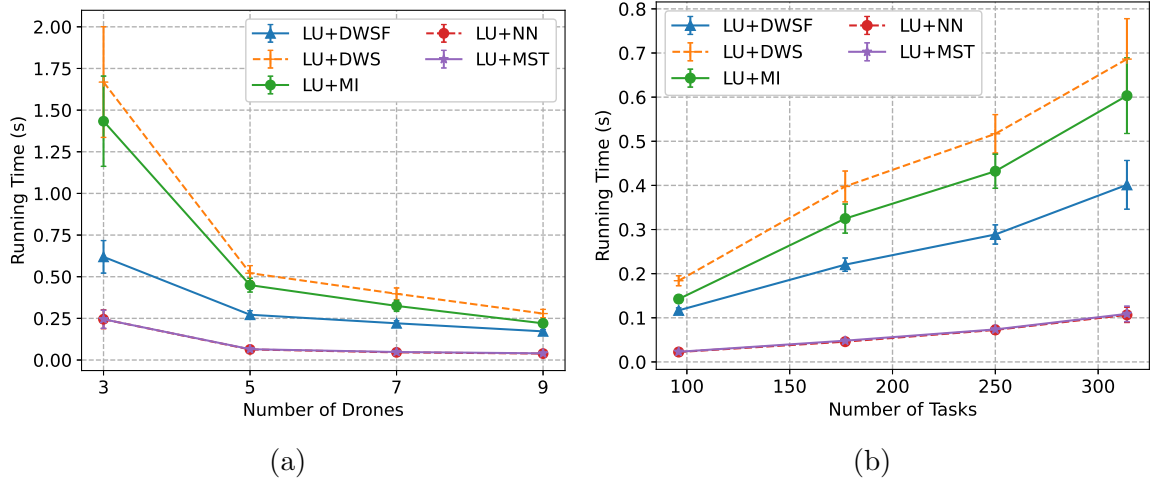


Figure 5.10: Total running time of the LU and scheduling algorithms under different numbers of (a) drones and (b) tasks. Sample results from simulations with (a) 177 tasks and (b) 7 drones.

collected from an Intel i7 workstation. This figure 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. 5.10b, we also observe that the running time of our LU and DWSF algorithms increases linearly to the number of tasks. Hence, Fig. 5.10 depicts that our LU and DWSF algorithms react fast even with larger problem size.

## 5.7 DragonFly Implementation

In this section, we present the implementation of our DragonFly system. We begin by describing the system’s structure in detail and then proceed to explain the implementation of each component. Furthermore, we conduct tests of our proposed waypoint scheduling algorithms in a lab-based testbed using a real drone. We compare the results with baseline algorithms to demonstrate the superior performance of our proposed approach.

### 5.7.1 System Architecture

The DragonFly system, as shown in Fig. 5.11, utilized drones equipped with RGB cameras to fly over high-rise fire settings, collecting data, which is then transmitted to the Incident Command Site (ICS) server. The ICS server tasks charge of data analysis, flight planning, and offers a dashboard for visualizing fire status and task assignment. The data manager

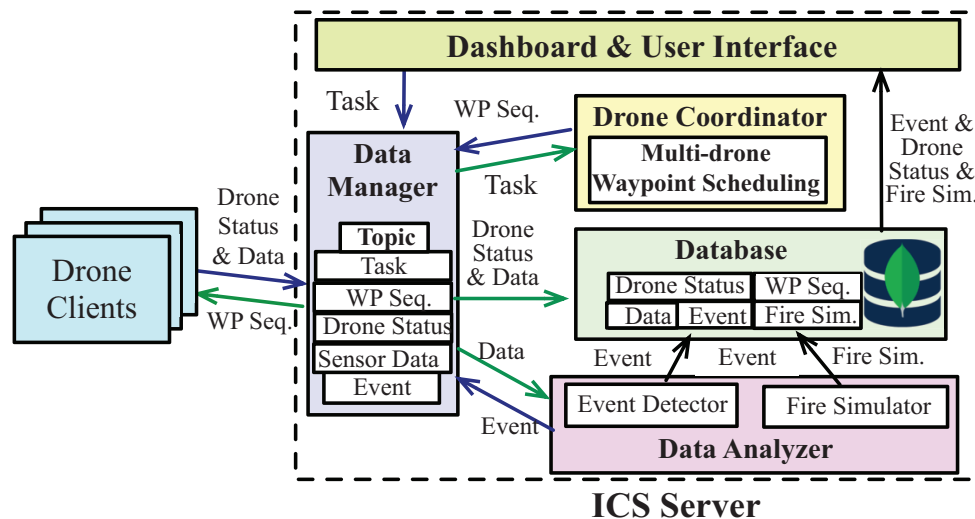


Figure 5.11: System Structure of DragonFly System.

continuously processes this data to reflect the current situations through the dashboard. The dashboard provides a user interface, enabling firefighters to specify monitoring tasks for the drones. Together with the building structure, these monitoring tasks are sent to the drone coordinator. The drone coordinator employs the AMT and DWSF algorithms to generate precise waypoint schedules for each drone. These waypoint sequences are then transmitted back to the data manager, which in turn sends them to the drone client. The drone client is responsible for commanding the flight of the drones. Each drone periodically reports the drone status, including GPS location and battery levels, along with images captured at each waypoint. The data manager efficiently receives this data and then forwards the drone status data to the Dashboard, providing real-time updates on the drone's activities and current state. Additionally, the captured images are transmitted to the data analyzer.

This system leverages the power of ZeroMQ, a robust pub/sub module, to facilitate data communication. This allows drone clients to publish drone status and sensor data, while the data manager efficiently subscribes to this valuable information. The data analyzer works for analyzing the received images and promptly reports any detected events such as detected fires or humans. This vital information is then rapidly updated on the dashboard, ensuring that firefighters have access to the latest status of the high-rise fire scenario.

### 5.7.2 Prototype Implementation and Experiments

**Lab-based testbed.** Due to the inability to participate in real high-rise fire settings, we created a lab-based testbed (see Fig. 5.12) and system prototype to evaluate the implementation and development challenges in DragonFly. We developed a mockup high-rise fire setting, comprising a scaled building facade measuring  $4 \times 5$  meters, featuring 8 windows. Two of these windows were equipped with simulated fires, while some of the others were adorned with curtains. We employ a DJI Tello drone, equipped with an RGB camera, to capture images within our testbed. The drone’s motion is controlled by a laptop drone controller, which receives waypoint sequences from an edge server and utilizes DJI SDK commands for navigation. All captured data is transmitted to the edge server via a WiFi network.

We have implemented the ICS server on a Windows 10 laptop, hosting all components. Additionally, we have set up a MongoDB-based database for long-term data storage in a separate server on Ubuntu 20.04. To ensure efficient data transmission, we utilize the ZeroMQ publish/subscribe model in the drone client to publish pictures captured by drones. On the receiving end, the data manager subscribes to receive the data and forwards it to the database and data analyzer. The data analyzer module integrates multiple image processing approaches to detect various critical events in high-rise fires, including human presence, and open window status using approaches in [67].

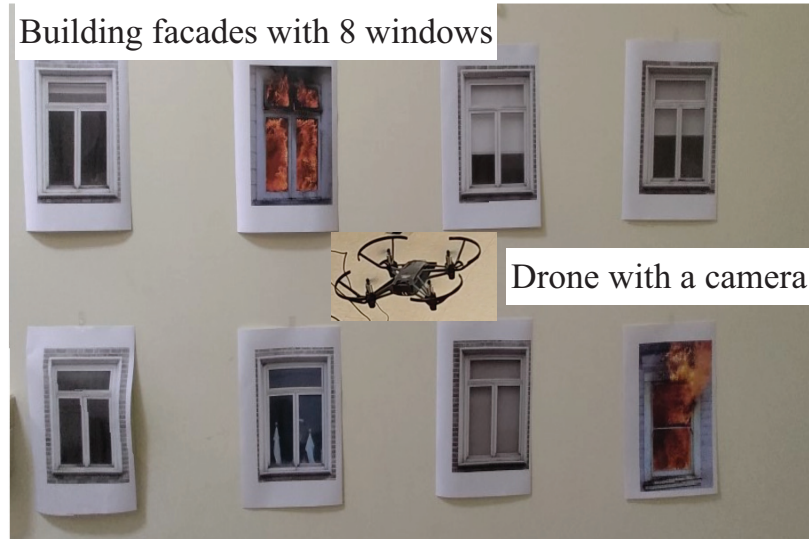
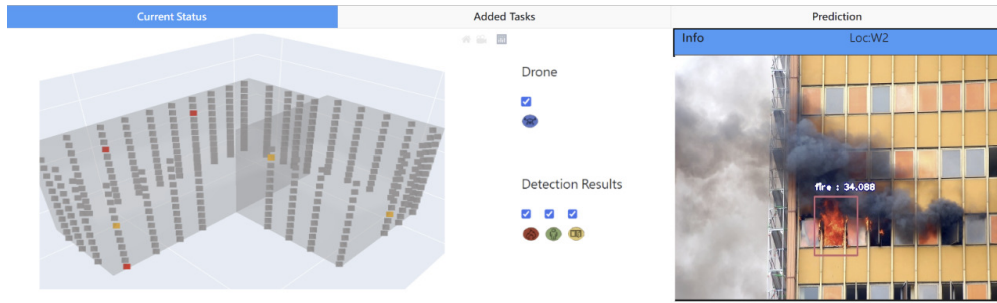


Figure 5.12: Mock-up Building Facade with Fires

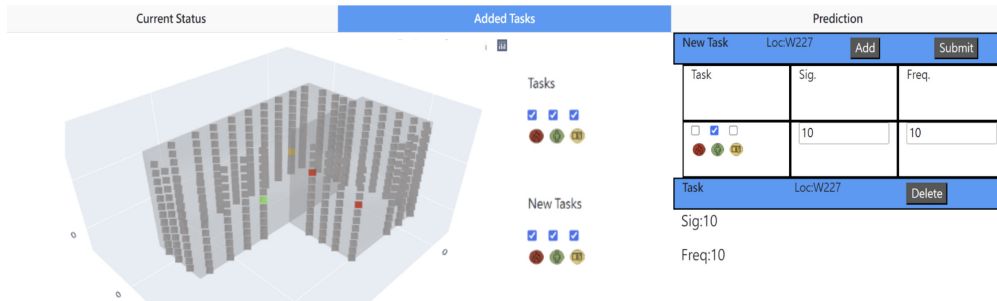
**DragonFly dashboard.** Moreover, we have developed the DragonFly dashboard using dash 2.5.1, enabling users to track the dynamic status of high-rise fires, monitor the drone’s position, and input tasks for mobile sensing. The dashboard includes three main pages as shown in Fig. 5.13. The Current Status page displays real-time fire scene state on the window box and the drone’s position around the building. The Add Tasks page enables users to add, modify, and remove tasks by directly selecting the window box, providing the task attributes, and clicking the button. The Prediction page provides users with the ability to view the current fire status and future fire predictions generated from the Fire simulator, which is based on the last fire events. Our testing environment includes Python 3.9, djitellopy 2.4.0, paho-mqtt 1.6.1, pyzmq 22.3.0, MongoDB 5.0.9, and Mosquitto 1.6.9.

**Real Drone Testbed Experiments.** We performed a series of experiments in our testbed, utilizing real drones, to evaluate the effectiveness of our proposed flight planning algorithm. We simulated multiple high-rise fires in our mockup fire setting. To create a realistic scenario, each window had a 20% chance of being opened and occupied. Additionally, we randomly simulate fires in these windows and generate tasks using the rules described in Section 6.6.1. Then, we planned the drone’s flight to fulfill those tasks using our DWSF algorithm and

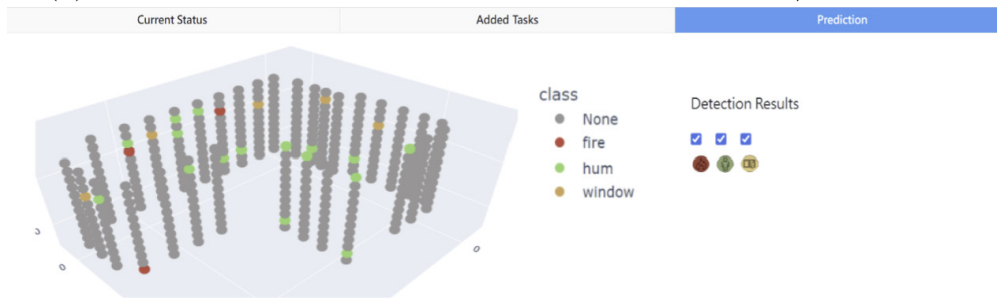




(a) Screenshots of the dashboard, which provide show the current status of high-rise building



(b) Screenshots of the dashboard, which provide UI to add/delete tasks.



(c) Screenshots of the dashboard, which show the future prediction of the high-rise fire

Figure 5.13: Dashboard of DragonFly System

several baseline methods, namely NN, MI, and MST, which were introduced in Section 5.6, to compare their performance

During the experiments, we set the speed of the drone as a constant value of 10 cm/s and conducted a comparison between the actual flight time and that of the simulated drone. The results, along with the real drone's trajectory, are depicted in Fig. 5.14. Our findings reveal that there are indeed some differences within 2 seconds in the speed of real drones and our simulated drone, especially when the flight distance is short and the speed is high. To

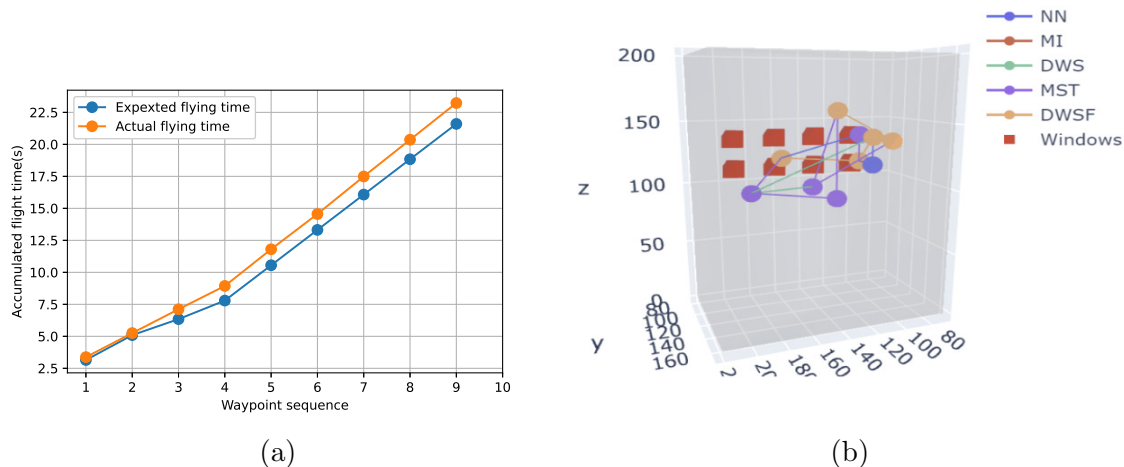


Figure 5.14: Real implementation with one fire source: (a) Expected and real fly time (b) Real trajectory with different algorithm

evaluate our real-world implementation of the DWSF flight planning algorithm, we compare its performance with other baseline algorithms using several metrics, including Minimum weighted AUC, Weighted accuracy, and Weighted reliability, as proposed in Section 5.6. Additionally, we present the simulated results, assuming the drone follows a constant speed during flight. The results, illustrated in Fig. 5.15 demonstrate the the superior performance of our DWSF algorithm across the three metrics when compared to other algorithms. Notably, we observe minimal errors in both real-world flights and expected outcomes. These promising results serve as strong validation of the efficacy of our approach and highlight its potential for practical application in high-rise fire monitoring scenarios.

## 5.8 Summary and Discussion

In this chapter, 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

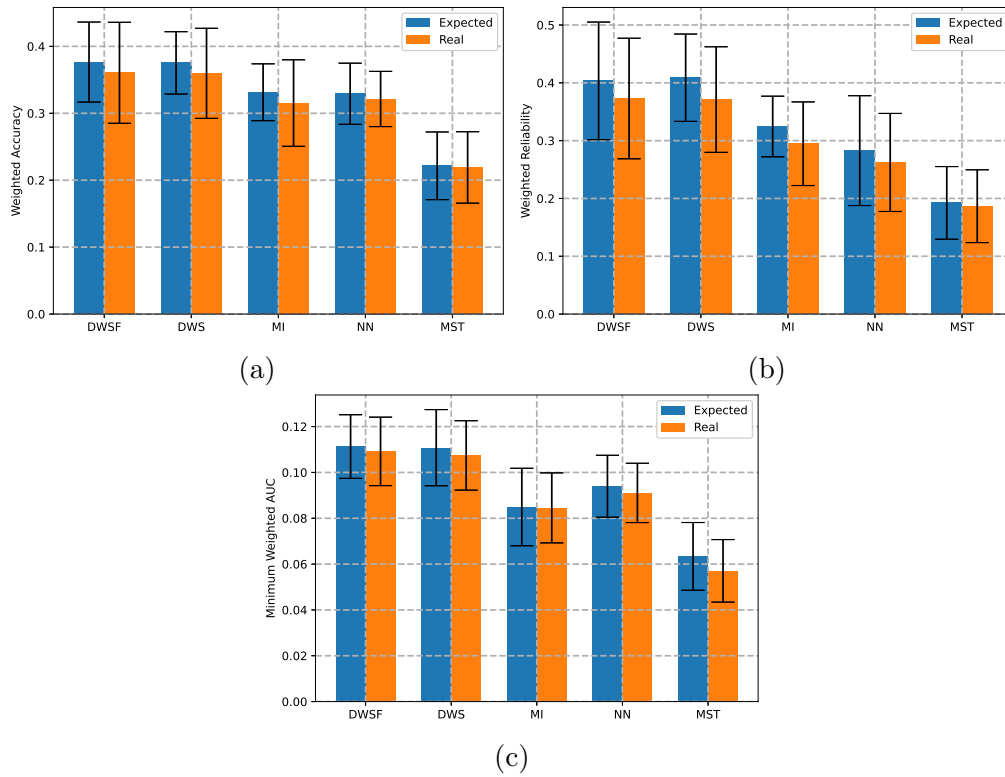


Figure 5.15: Performance of our DWSF Algorithm in Real Implementation on (a) Weighted Accuracy, (b) Weighted Reliability, and (c) Minimum Weighted AUC.

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. In the upcoming chapter, our goal is to minimize human intervention in drone-based monitoring procedures by enhancing system automation. To achieve this, we intend to establish a control loop within the IoT system by integrating vision processing techniques. This integration will enable the system to perceive the dynamic status of the physical environment in real time. We plan to utilize formal logic to design rules that represent high-level human monitoring requirements. Additionally, we will develop a multi-drone flight planning procedure to guide drones in capturing data for monitoring large-scale extreme conditions, even in areas with poor network conditions like wildfire monitoring. Our strategy involves empowering drones to store data onboard when they lose network connectivity and upload it when the network connection is reestablished, particularly in such scenarios. Our ultimate aim is to optimize the flight planning of drones, thereby improving both the quality of captured data and sensor coverage.

## Chapter 6

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

This chapter further explores the integration of on-demand drone mobility into IoT systems for monitoring emergent events, aiming to improve both sensing and network capabilities. We introduces DOMÉ, a drone-assisted monitoring system specifically designed to gather real-time data for monitoring emergent and evolving events, with the goal of enhancing situational awareness. Our driving use case is the prescribed burn event (Rx fire), which involves man-made wildland fires used to reduce hazardous fuels in forests. DOMÉ 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. DOMÉ 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).

## 6.1 Chapter Overview

This chapter 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 [202]. From 2012 to 2021, the US had an average of 61,289 wildfires, impacting  $\sim 7.4$  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 [8]. 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 [48]. While helpful, these approaches have practical limitations, such as delays, coarse data resolution and maintenance difficulties due to fire damage [202]. 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 [69, 158] 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 work, 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 [21]. 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 [223] 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 chapter are: i) the design of DOME, a drone-based monitoring system (§6.2), 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 (§6.3), iii) formulation of the multi-drone flight planning (MFP) problem; and a two-step approach to solve it effectively (§6.4,§6.5), iv) implementation of the DOME prototype and thorough evaluation of our proposed algorithms in Rx burn use cases (§6.6).

## 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 6.1. 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





Figure 6.1: Our Previous Rx Burns.

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 [129]. Similarly, improving RGB image resolution allows us to improve object detection results with more granular object recognition outcomes [3]. 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 [234]. These

communication technologies vary in data transmission range from 200 to 3000 meters [202]. 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 [132] 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).

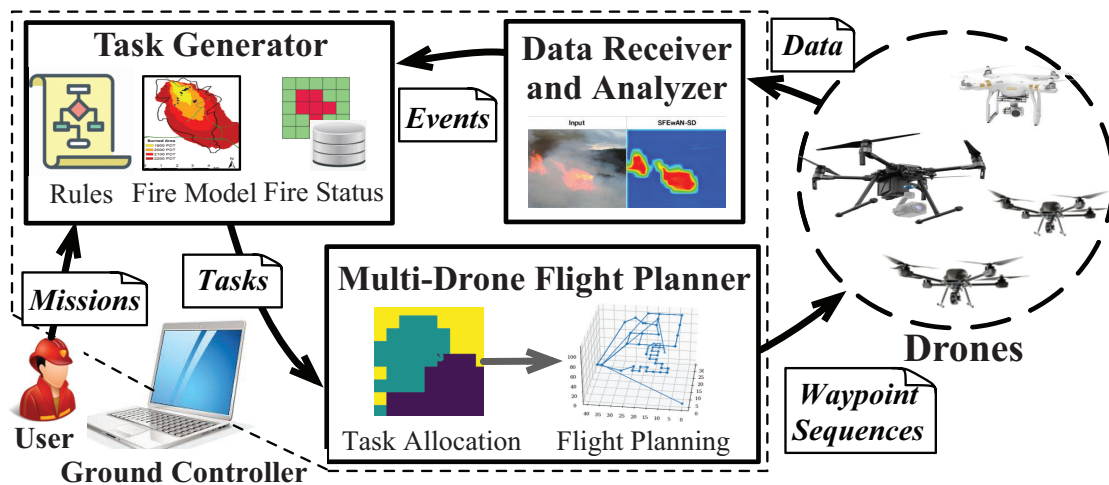


Figure 6.2: Overview of DOME Framework.

**Existing work and limitations.** Next, we expand upon the related work in Section 2.4, offering a more detailed examination of drone-based mobile sensing from a technical perspective. We assess related efforts in multi-drone planning and drone-based monitoring, which have been explored in many fields, including operations research (OR) [142] artificial intelligence (AI) [211], and robotics [50]. 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 [142]. Here, popular methods such as branch-and-cut and dynamic programming have been utilized by CPLEX [52] to obtain optimal solutions, but they are computation-

ally expensive and intractable. This has led to a rise in heuristic-based methods, such as tabu search [10], genetic algorithms [66], evolutionary algorithms [161], and two-phase methods [125], which solve VRPs fast, but with suboptimal solutions. To add to the complexity, domain-specific monitoring requirements and other constraints in Rx Fires 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 [211]. The state-of-the-art solutions involve modeling MAP instances using Planning Domain Description Language PDDL or Multi-agent PDDL [7], and applying PDDL solvers such as ADP [54] 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 [60] to horizon optimization and model predictive control for cinematography [153]. Other efforts have leveraged auction-based task allocations [221], Markov decision processes [152], and Monte-Carlo tree search [150] 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 [36]. Subproblems considered in this regard involve the efficient decomposition of non-uniform areas [184], heuristic algorithms [113] for area partition among multiple UAVs, and waypoint generation and path planning for coverage [19]. 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.

Drone-centric approaches have been explored in many applications showing strong potential in enabling real-time disaster monitoring applications, such as building fire monitoring [135], military missions [23] and environmental monitoring [177]. These settings often require drones to monitor target areas with dynamic threats, and hostile environments [127]. In the context of wildfires, the goals of such monitoring include tracking the fire perimeter [77] and fire intensity deviation [168]. 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 work proposes DOME, a system to support multi-drone flight planning for monitoring Rx burns, as shown in Fig. 6.2. 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 (BM)**: localizing firefighters and checking the state of equipment at the burn site; ii) **Fire detection (FD)**: detecting fires; iii) **Fire tracking (FT)**: tracking fire spread near the fire front; and iv) **Fire intensity inspection (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 (FI) every 3 minutes or prioritize fire

tracking (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. 6.2) 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. 6.2), 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 the 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 the 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 utilizes 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.

### 6.3 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 §6.2: Fire detection (FD), Burn site resource monitoring (BM), Fire tracking (FT), and Fire intensity inspection (FI). Fig. 6.3 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 [68]) and iii) a rule engine with user-specified rules and a rule interpreter (PyKnow [4]) 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 ' $\Rightarrow$ ,' 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  $\mathbf{G}$  into multiple non-overlapping square grids and track the state of each grid cell  $g \in \mathbf{G}$ . The state

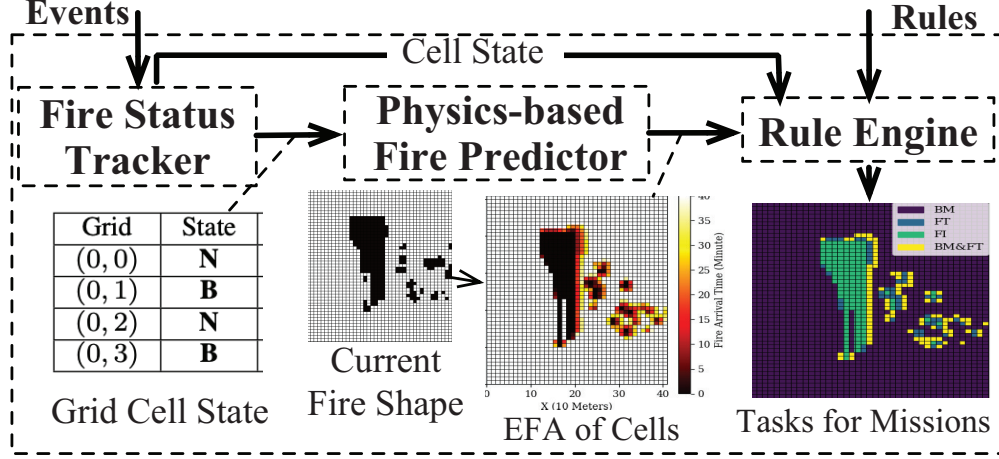


Figure 6.3: Workflow of Task Generator.

$s$  of each  $g \in \mathbf{G}$  is one of  $\{\mathbf{UK}, \mathbf{B}, \mathbf{NB}, \mathbf{BO}\}$ , 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. 6.4. 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 [68], 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 ig-

nition 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 a 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 **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 **FI** for checking the fire intensity, **BM** for monitoring human/equipment status, and **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. 6.4. 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  $m \in \{\mathbf{BM}, \mathbf{FD}, \mathbf{FT}, \mathbf{FI}\}$  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  $s \in \{\mathbf{UK}, \mathbf{B}, \mathbf{NB}, \mathbf{BO}\}$ .
- $Fire(g, t)$ : 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 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  $s \in \{\mathbf{UK}, \mathbf{B}, \mathbf{NB}, \mathbf{BO}\}$  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 6.4: Task Generation Rules

Rule **RT-1** generates tasks for mission  $\mathbb{FD}$  to cover all grid cells with state  $\mathbf{UK}$  at the start of an epoch. We use the symbol ' $\sim$ ' 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  $\mathbf{UK}$ . 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  $\mathbf{B}$ , and Rule **RT-5** generates a task for mission  $\mathbb{BM}$  at each cell under state  $\mathbf{NB}$  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.6.5, to support the dynamic update of ongoing tasks during the epoch; meanwhile, drones can adjust their flights to accommodate these updates. The actions

---

**Facts:**

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

**Actions:**

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

**Rules:**

**RE-1** :  $\forall g(\neg \text{Monitor} \wedge (\text{State}(g, t) = \mathbf{UK}) \wedge (\text{Fire}(g, t) \vee \text{NoFire}(g, t)) \Rightarrow \text{Delete}(\text{FD}, g, t))$

**RE-2** :  $\forall g(\text{Monitor} \wedge (\text{State}(g, t) = \mathbf{NB}) \wedge \text{Fire}(g, t) \Rightarrow \text{Delete}(\text{FT}, g, t) \wedge \text{Add}(\text{FI}, g, t, \sim))$

**RE-3** :  $\forall g(\text{Monitor} \wedge (\text{State}(g, t) = \mathbf{B}) \wedge \text{NoFire}(g, t) \Rightarrow \text{Delete}(\text{FI}, g, t))$

**RE-4** :  $\exists g(\text{Monitor} \wedge (\text{State}(g, t) = \mathbf{NB}) \wedge \text{Fire}(g, t) \wedge (t \leq \text{EFA}(g) - \delta_{an})) \Rightarrow \forall g(\text{GetEFA}(g, t)) \wedge \text{Anomaly}(t + 1)$

**RE-5** :  $\forall g(\text{Anomaly}(t) \wedge (\text{State}(g, t) = \mathbf{NB}) \Rightarrow \text{Update}(\text{BM}, g, t, \max\{t, \text{EFA}(g) - \delta_{ft}\}) \wedge \text{Update}(\text{FT}, g, \max\{t, \text{EFA}(g) - \delta_{ft}\}, \sim))$

---

Figure 6.5: Rules for event-driven task update

$\text{Delete}(m, g, t)$  and  $\text{Update}(m, g, st, et)$  are used to remove and update the time duration of tasks, respectively. Before the monitoring phase, rule **RE-1** deletes tasks for **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 **FI** and **FT** during monitoring when fires are detected/dissipated. Since fires may deviate from expected propagation plans, we define an 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.,  $t \leq \text{EFA}(g) - \delta_{an}$ , 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 **BM** and **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.

The successful implementation of the event-driven task update procedure requires uninterrupted communication between the Ground Control (GC) and the drones, as well as among the drones themselves. This ensures constant updates of tasks during drone flights and facilitates task reallocation among drones as needed. However, due to the possibility of network disconnection during the drones' flights in our scenario, this study is restricted to exclusively focusing on utilizing a time-driven task generation procedure. Tasks are provided to drones at the start of each epoch when they return to connect with the GC, ensuring a reliable task allocation process.

## 6.4 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.

### 6.4.1 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  $\sigma(j)$ .

We guide drones to capture valuable high-resolution data by evaluating and scoring data quality for diverse missions using Pixel Per Meter (PPM) measurements. We compare the resolution against threshold values proposed in empirical standards, including pixel density requirements for detection, observation, and recognition in CCTV systems [3], and Johnson’s criteria [129] for thermal images’ PPM requirements for detecting, recognizing, and identifying objects based on target dimensions (height  $\times$  width) of human and fire ( $1 \times 0.5m$  and  $1m \times 1m$  respectively).

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) measurements. We compare the resolution against threshold values proposed in empirical standards, including RGB pixel density requirements for detection, observation, and recognition in CCTV systems [3], and Johnson’s criteria [129] for thermal images’ PPM requirements for detecting, recognizing, and identifying objects based on target dimensions (height  $\times$  width) of human and fire ( $1 \times 0.5m$  and  $1m \times 1m$  respectively). 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 6.1 shows a data quality scoring rubric. For example, the cell located at the intersection of the Thermal and  $\mathbb{B}\mathbb{M}$  represents that if a thermal camera captures data for  $\mathbb{B}\mathbb{M}$  with PPM value  $pm$ , then the data’s quality score is 0 if  $pm < 8.48$ , 0.6 if  $8.48 \leq pm < 10.6$ , 0.75 if  $10.6 \leq pm < 15.2$  and 0.9 otherwise. As RGB images cannot reveal fire intensity (temperature) information, we set ‘N/A’ in the cell for RGB and  $\mathbb{F}\mathbb{I}$ .

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

Table 6.1: Data quality score under diverse PPMs

Sensor \ Mission	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				

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 work, 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. (6.1a) and (6.1b) derived in [57] 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.

$$PPM_s(h) = PH_s / (2h \times \tan(FH_s/2)) \quad (6.1a)$$

$$CR_s(h) = 2h \times \min \{ \tan(FH_s/2), \tan(FV_s/2) \} \quad (6.1b)$$

**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 6.1. Then, we deduce the set of potential heights at which sensor  $s$  should capture data as  $\mathbf{HSet}(j, s) = \{\min(h, H_{max}) : PPM_s(h) \in \mathbf{TH}(j, s), h \geq H_{min}\}$ . 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.,  $rng_d \geq Dis(w_i, g')$  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.

## 6.4.2 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. (6.2) which evaluates the value of data that drone  $d$  captured at  $w_j \in \mathbf{Q}^d$  for task  $T_i \in \mathbf{T}$ .

$$V_d(w_j, T_i) = \sigma(m_i) \times \max_{s \in sen_d} (SC_s^{m_i}(PPM_s(z(w_j))) \times Cov_s(w_j, T_i)), \quad (6.2)$$

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.

### 6.4.3 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

$$\mathbb{R}(\mathbf{Q}, T_i^k) = \begin{cases} -\beta, & \text{if } \sum_{d \in [1, D]} |\Theta(\mathbf{Q}^d, T_i^k)| = 0, \\ \max_{d \in [1, D]} \left( \max_{n \in \Theta(\mathbf{Q}^d, T_i^k)} V_d(w_{q_{(n)}^d}, T_i) \right), & \text{else.} \end{cases} \quad (6.3)$$

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 work, 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:



$$\max_{\mathbf{Q}} \sum_{T_i^k: t_0 < d(T_i^k) \leq t^*; T_i \in \mathbf{T}} \mathbb{R}(\mathbf{Q}, T_i^k) \quad (6.4a)$$

$$\text{s.t.} \quad AT(\mathbf{Q}^d, |\mathbf{Q}^d|) \leq t^*, \quad \forall d \in [1, D], \quad (6.4b)$$

$$w_{q_{(0)}}^d = w_0^d, \quad w_{q_{(\mathbf{Q}^d)}}^d = w_{dpt}, \quad \forall d \in [1, D]. \quad (6.4c)$$

Our objective function (Eq. (6.4a)) 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 (6.4b) requires that each drone's flying time does not exceed the end time of the epoch, while constraint (6.4c) 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. (6.4c) 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 [81] to a special case of our problem, where there is only one mission with a single PPM requirement and a single drone.

## 6.5 Proposed Algorithms for MFP

We solve the MFP problem in two steps and design heuristics as follows: i) task allocation (§6.5.1), which assigns every generated task to a specific drone; and ii) flight planning (§6.5.2), where each drone computes its waypoint sequence to fulfill its given tasks.

### 6.5.1 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. (6.4a)) 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

$$\mathbb{U}_d(\mathbf{T}') = \sum_{j=1}^M ET_d(\{T_i : T_i \in \mathbf{T}', m_i = j\})/p(j) \quad (6.5)$$

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, as shown in Alg. 5, 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, as shown in Alg. 6. 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 [70], 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  $\mathcal{O}(|\mathbf{W}_c|^2)$  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  $\mathcal{O}(|\mathbf{T}|^2 D |\mathbf{W}_c|^2)$ .

---

**Algorithm 5:** Utilization-based task allocation (UTA)

---

**Input:** Tasks  $\mathbf{T}$ , Drones  $D$ , Burn site area, square size for task clustering  $\Gamma$

**Output:** Task allocation of  $\mathbf{T}^* = \{\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_D\}$

```
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  $D_i \leftarrow \mathbf{D}.groupby("sensor")$  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  $\mathbb{U}_d$   
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  $\{\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_D\}$ 
```

---

---

**Algorithm 6:** 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  $T_i$ 's mission  $m_i$  then return  $+\infty$ ;
   // Get the WPC drone  $d$  can execute  $T_i$  and has maximum coverage.
3   Get  $s' \leftarrow \arg \max_{s \in \text{sen}_d} CR_s(H(s))$  with
    $H(s) = \max\{h : PPM_s(h) \geq \min(\mathbf{TH}(m_i, s)), h \in \text{AllHeights}(\mathbf{W}_d)\}$ 
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  $U_d(\mathbf{T}')$ 
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  $\sum_{k=1}^M ET_d(\mathbf{T}'_k) / p(k)$ 
```

---

### 6.5.2 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. (6.2)), 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  $\mathbb{T}\mathbb{X}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w') = \langle w', t', \mathbf{S}'(\mathbf{T}_d) \rangle$  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  $\phi_i \leq t'$ , 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*  $\mathbb{A}\mathbb{R}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$  as the overall change of rewards for subtasks (specified in Eq. (6.3)) 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  $K = |\{T_i : T_i \in \mathbf{T}_d, t < DL_i < t', UR_i = 0\}|$ , 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  $w'$ . Alg. 8 presents the pseudo-code for state transition and computing each action reward.

Each episode in our MDP starts with the initial state and ends when it reaches a state with  $t \geq t^*$ . 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.

**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.,  $t'' \leq \min\{DL_i : T_i \in \mathbf{T}_d, SR_i > 0\}$  and early enough to return in time, i.e.,  $t'' + FLT_d(\mathcal{UP}(w'), w_{dpt}) \leq t^*$ . 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  $Que$ , and update it whenever certain subtasks are released or expired. In the algorithm, we use a flag  $STUupdate$  to track the tasks’ update; once its subtasks are newly released, the  $STUupdate$

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 ( $STUpdate = \text{True}$ ), 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  $\mathcal{O}(\frac{t^* - t_0}{T_{lo_i}} |\mathbf{W}_d| |\mathbf{T}|)$ , where the  $\frac{t^* - t_0}{T_{lo_i}}$  indicates the times of waypoint selections when drone stays static. Algs. 7 and 8 shows DFP's detailed pseudo-code.



---

**Algorithm 7: 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  $STUupdate \leftarrow False$ ;  $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle \leftarrow State_0$ ,  $\mathbb{R}_d \leftarrow 0$ ;  
2 Group all subtasks based on their deadlines, sort them from  $d_i$  smallest to largest:  
    $Que = [\{T_i : T_i \in \mathbf{T}_d, DL_i = d_1\}, \{T_i : T_i \in \mathbf{T}_d, DL_i = d_2\} \dots]$ .  
3 while  $t \leq t^*$  do  
4   if  $STUupdate$  then Update  $Que$  for new released subtasks ;  
5    $Seq \leftarrow 0$  // Enter Fast Coverage phase.  
6   while  $Seq < |Que|$  and  $\neg STUupdate$  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  $\neg STUupdate$  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       $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUupdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$ ;  
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  $\neg STUupdate$  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        $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUupdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$ ;  
21        $\mathbf{Q}^d.add(w')$ ;  $\mathbb{R}_d \leftarrow \mathbb{R}_d + \mathcal{AR}$   
22   if (not  $\text{VALID}(UP(w))$ ) then  $w' \leftarrow w_0^d$ ; // Return back  
23   else  $w' \leftarrow UP(w)$  // Upload data to the GC.  
24    $\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUupdate \leftarrow \text{TRANS}(\langle w, t, \mathbf{S}(\mathbf{T}_d) \rangle, w')$ ;  
25    $\mathbf{Q}^d.add(w')$ ;  $\mathbb{R}_d \leftarrow \mathbb{R}_d + \mathcal{AR}$   
26 return  $\mathbf{Q}^d, \mathbb{R}_d$ .
```

---

---

**Algorithm 8:** State transition 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  $\mathcal{AR}$ ,  $STUupdate$

```
1  $t' \leftarrow t + FLT_d(w, w')$ ;  $\mathcal{AR} \leftarrow 0$ ;  $STUupdate \leftarrow False$ 
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      $k \leftarrow \lfloor (t' - DL_i) / p_i \rfloor$ ;  $\mathcal{AR} \leftarrow \mathcal{AR} - k\beta$ ;  $STUupdate \leftarrow True$ 
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  $\langle w', t', \mathbf{S}(\mathbf{T}_d) \rangle, \mathcal{AR}, STUupdate$ 
```

---

## 6.6 Experimental Evaluation

In this section, we evaluate our proposed algorithms by comparing them with several baseline algorithms and exploring their scalability in different simulated Rx fire scenarios.

### 6.6.1 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

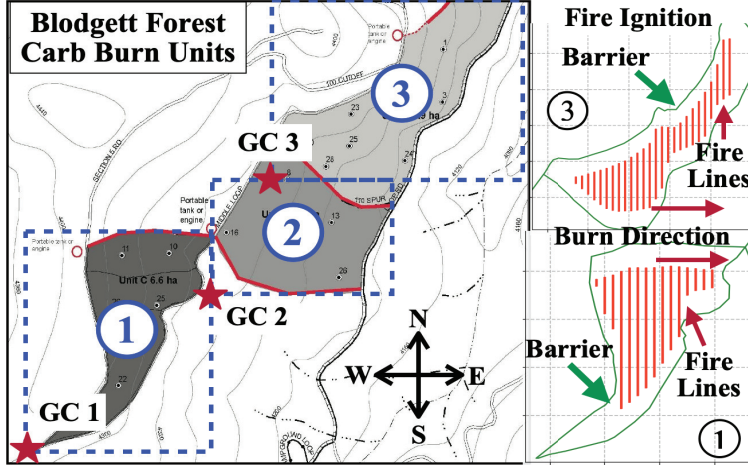


Figure 6.6: Illustration of three burn sites and fire ignition strategy.

Fig. 6.6. In our simulation, a sequence of fire stripes is added progressively, as shown in two sub-figures in Fig. 6.6. 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  $\mathbb{B}\mathbb{M}$ ,  $\mathbb{F}\mathbb{I}$ , and  $\mathbb{F}\mathbb{T}$  missions, using our proposed rules in §6.3 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 6.1. 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 6.2 lists more simulation parameters.

Our study records the following performance metrics: (i) *Total Reward*, as defined in the MFP's objective function in Eq. (6.4a); (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

Table 6.2: Simulation Parameters

Parameter	Value	Parameter	Value
Simulation time	80 min	Penalty $\beta$	10
Epoch length	20 min	Grid size	$10 \times 10m$
Wind direction	$270^\circ (\pm 30^\circ)$	Flying speed	5 m/s
Wind speed	5-25 ( $\pm 3$ )mph	Loiter time	2 s
Ignition interval	10 ( $\pm 3$ ) min	Drone number	4 – 16
Fire strip gap	10 ( $\pm 3$ ) 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 $(p, \sigma)$	BM(10 min, 2), FI(5 min, 1), FT(2.5 min, 3)		
Drones	DJI Zenmuse XT2 (XT2); DJI Air 2S (2S)		
Sensor FOV	XT2: thermal $45^\circ \times 37^\circ$ , RGB $57^\circ \times 42^\circ$ ; 2S: $72^\circ \times 58^\circ$		
Sensor Res.	XT2: thermal $640 \times 512$ , RGB $3840 \times 2160$ ; 2S: $5472 \times 3078$		

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) [113], 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) [70], a classical greedy algorithm for vehicle routing problems. We also compared it with an extended version of the Earliest-Deadline-First baseline algorithm [203], 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 §6.5.2) 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.

## 6.6.2 Experimental Results

**Comparative Performance.** Figs 6.8 and 6.9 compare the performance of the proposed UTA-DFP algorithm with baseline algorithms in burn sites ② and ③. We provide sample results from simulations with six drones (three of each type) under 10 mph wind conditions over four epochs, each lasting 20 mins. At the beginning of each epoch, we use the defined rules in §6.3 to generate tasks based on the simulated fire spread status. The number of generated tasks for diverse missions is shown in Fig. 6.7. 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 6.8 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 in burn site ②.

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. 6.11. Figs. 6.11a and 6.11b 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

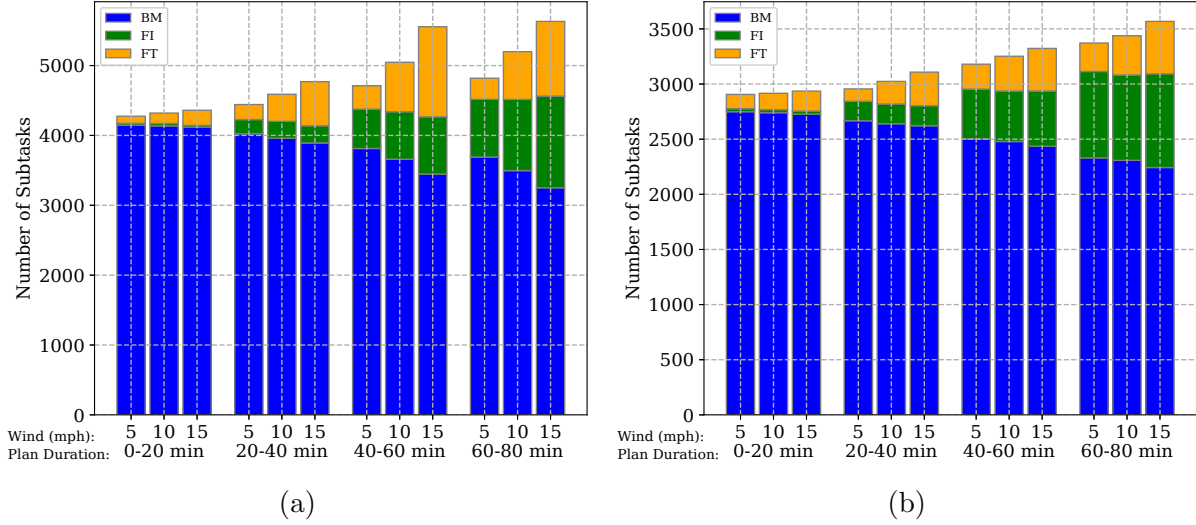


Figure 6.7: Subtask number in burn sites ① (a), ② (b) under diverse wind speeds

achieving 0.486% 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 FT and FI are generated. This reveals that UTA can handle task heterogeneity better than distance-based approaches.

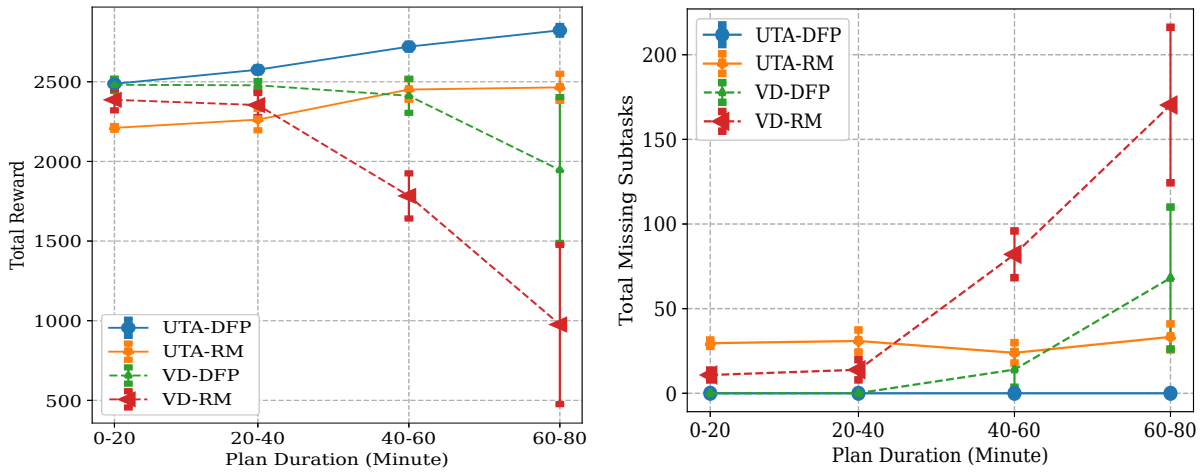


Figure 6.8: Performance of our UTA-DFP algorithm at Burn sites ②, (a) gives total reward and (b) gives total missing subtasks.

**Flight planning algorithms.** Next, we compare our proposed DFP algorithm with our baseline algorithms when using the UTA algorithm for task allocation in Figs. 6.11c and

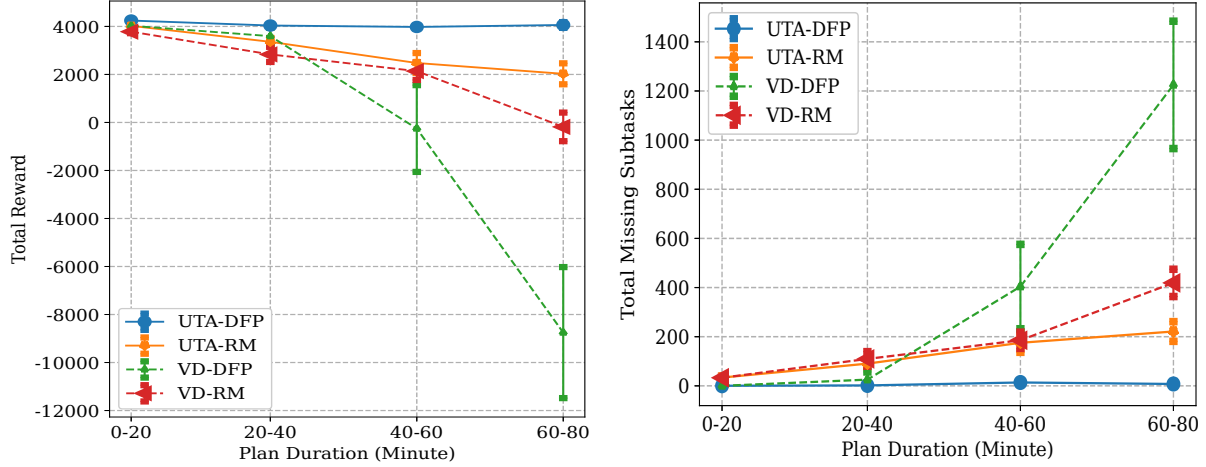
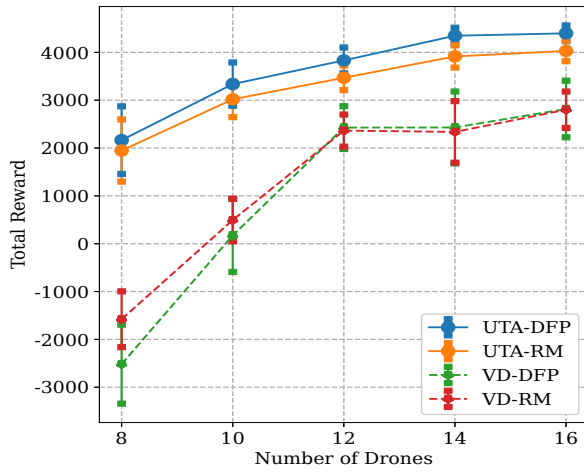


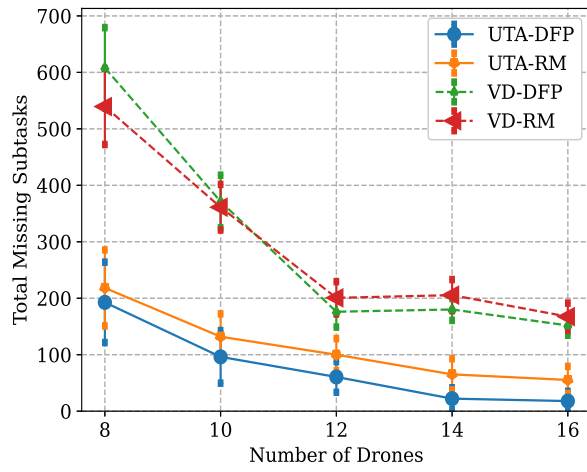
Figure 6.9: Performance of our UTA-DFP at Burn sites ③. (a) gives the total reward, and (b) the total missing subtasks.

6.11d. The results demonstrate *our DFP flight planning algorithm always leads to higher overall reward and fewer missing subtasks*, which achieves 0.16 times 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. 6.11c and 6.11d 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.

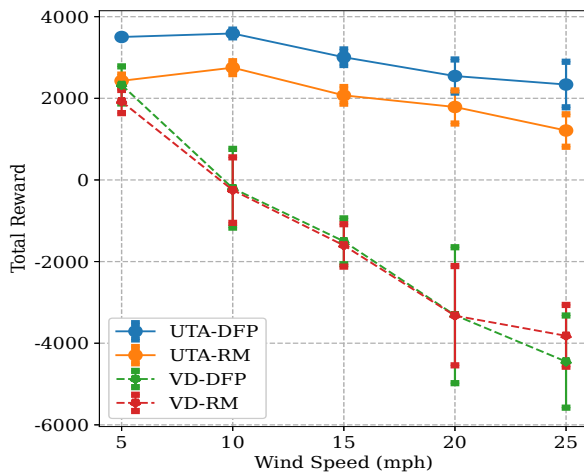
**Scalability of our proposed algorithm.** Next, we consider a larger MFP problem with a different number of drones (from 8 to 16) and tasks (from 4000 to 6000). Figs. 6.10a and 6.10b present the performance of these algorithms in burn site 1 during 60-80 min under 10 mph wind under a diverse number 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



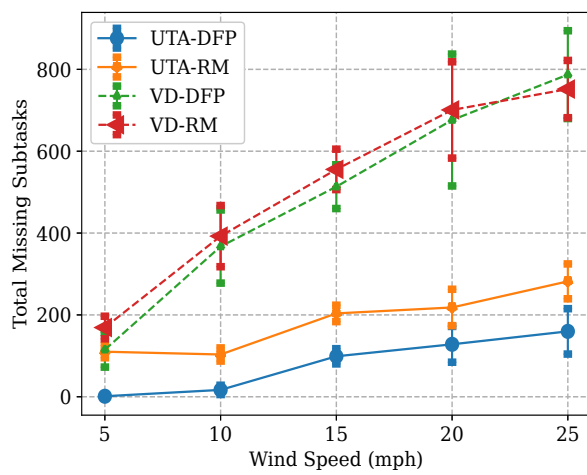
(a)



(b)



(c)



(d)

Figure 6.10: Performance of (a), (b) task allocation and (c), (d) flight planning algorithms. (a), (c) give the total reward, (b), (d) give the missing subtasks.



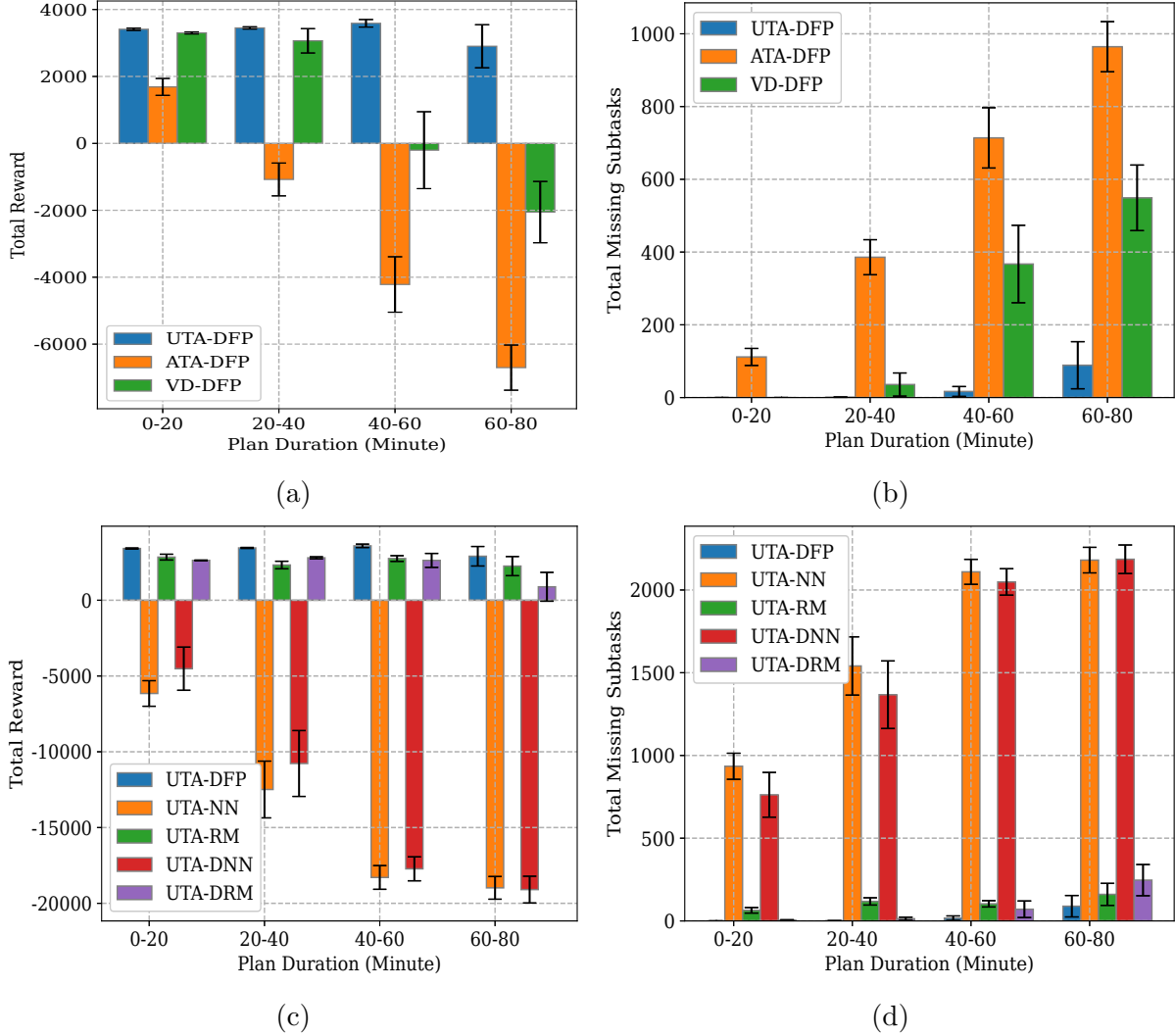


Figure 6.11: Performance of (a), (b) UTA and (c), (d) FDP algorithms. (a), (c) give the total reward, (b), (d) give the missing subtasks.

the highest total reward and fewer missing subtasks, which achieve 0.483 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. 6.10c and 6.10d 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.6.7a, 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.

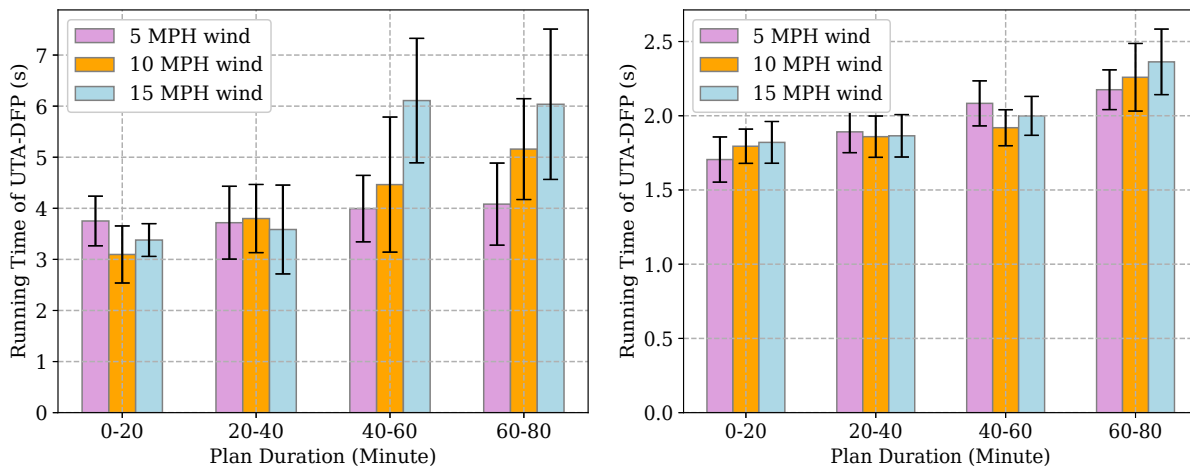


Figure 6.12: Running time of UTA-DFP at burn sites ① (a) and ② (b)

Fig. 6.12 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.

## 6.7 System Implementation

We have implemented the DOME system, integrating in-situ sensor data to provide environmental information related to wind conditions and air quality. These weather conditions would be utilized for fire prediction and air pollution monitoring. Next, we will introduce the detailed system architecture and illustrate our testbed.

Our DOME system, depicted in Figure 6.13, utilizes drones equipped with RGB cameras and various in-situ sensors to collect real-time data for monitoring critical factors during wildland fires, such as fire status, air quality, and wind conditions. All collected data is transmitted

via WiFi networks to an edge server, which is responsible for controlling the drone-based sensing process and managing data integration, analysis, and storage. The DOME code and detailed installation and implementation instructions are publicly available in [5]. Next, we will illustrate the DOME system structure by describing its four primary functions.

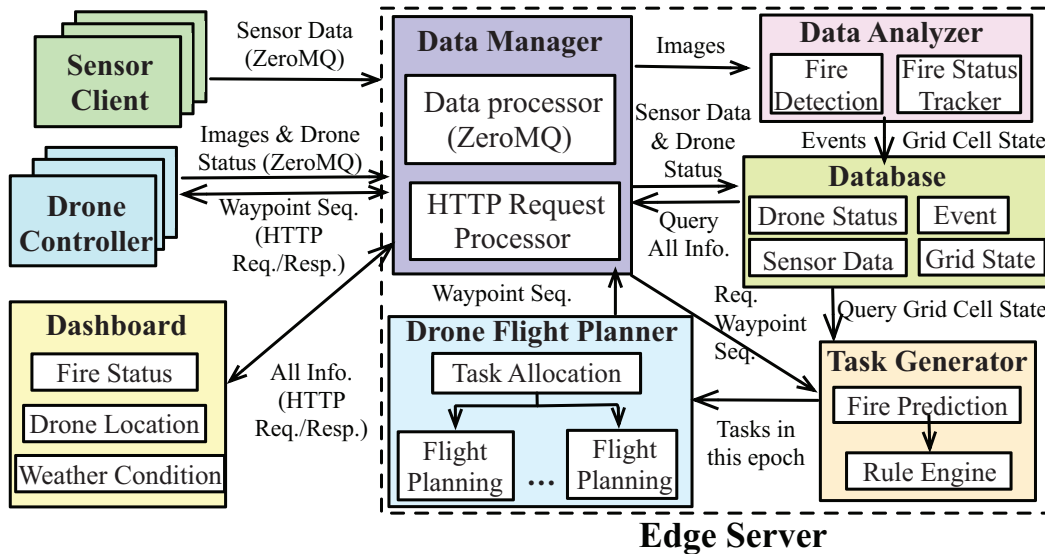


Figure 6.13: System Architecture of DOME

**In-situ sensor data collection.** As shown in Figure 6.14, the DOME system incorporates in-situ sensors placed near the fire setting, which periodically collect and transmit sensing data to the edge server. Our system employs three types of sensors for monitoring: wind (speed and direction), weather (temperature and relative humidity), and PM (particulate matter for smoke monitoring). To facilitate the data collection process, each sensor is connected to a Raspberry Pi (RPi) through various hardware communication interfaces. The weather sensor is connected to the RPi via GPIO, the wind sensor is connected through RJ11 cables, and the PM sensor communicates with the RPi through a USB interface. All RPis are running Scale-Client [29], a modular software stack that collects raw data from sensors and extracts sensing data by parsing them. These in-situ sensors capture data at 30-second intervals, which is then transmitted and stored in our edge server, a laptop, through a local WiFi network. For efficient communication, we utilize the Zero Message Queue (ZeroMQ), an

open-source publish and subscribe communication model, to send and receive various types of sensing data in parallel. Within each RPi, a publisher is created for each sensor, and it publishes data with a specific topic corresponding to its sensing type and ID following a pre-defined period. On the edge server, a **data manager** component subscribes to all sensing data and stores them in a MongoDB-based database.



Figure 6.14: Sensors, Testbed and Dashboard in DOME

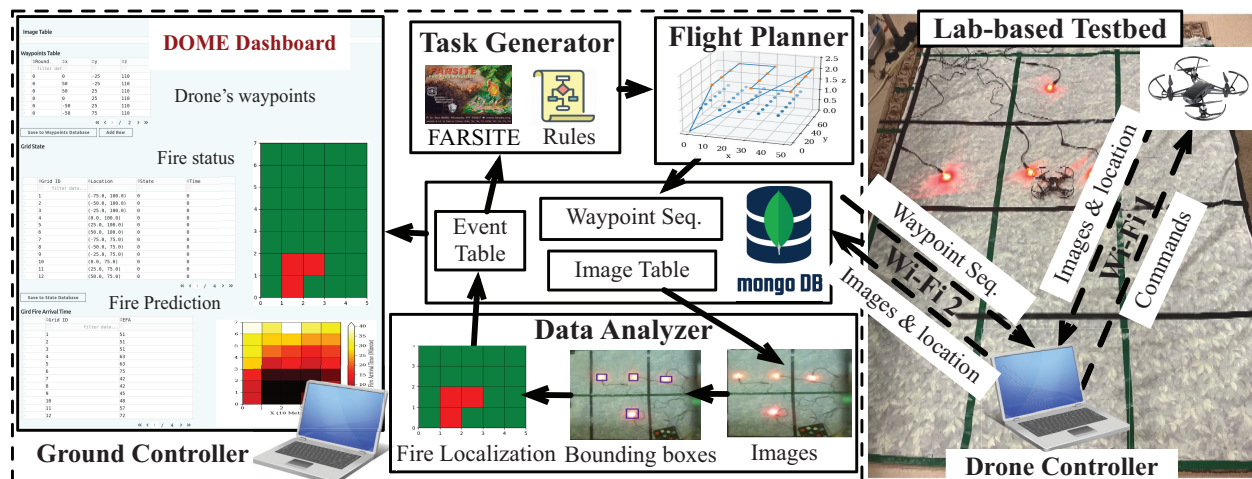


Figure 6.15: Drone-based Mobile Sensing in DOME System

**Drone-based mobile sensing.** For data collection and flight control, we utilize a DJI Tello drone equipped with an RGB camera and the Tello SDK interface (see Fig. 6.15). A

laptop **drone controller** sends commands to direct the drone to fly above the fire setting following a predefined waypoint sequence, capturing top-down view RGB images at each waypoint. These images are time and location-stamped and transmitted to the edge server by the drone controller using ZeroMQ via a WiFi network. The **data manager** receives the images and passes them to a **data analyzer**, which employs image processing techniques such as FireNet [93] to detect and locate fires. The fire setting is divided into grid cells to represent the fire status, categorized as 'Fire' or 'No fire.' Each fire detection result, along with the fire location (grid cell) and detection time, is treated as an event to update the state of the grid cells. The rules governing state transitions are described in Section 6.3.

**Drone flight planning.** The DOME system schedules drone flights periodically for predefined time periods called epochs, each lasting three minutes. At the beginning of each epoch, each drone controller sends a waypoint query HTTP request to the data manager, which handles the request using Flask. The data manager then triggers the **task generator** to create tasks specifying the spatial and temporal requirements for drone data collection based on the fire status, predicted fire evolution, and user-defined task generation rules (see Section 6.3). The **drone flight planner** then assigns these tasks to specific drones and generates waypoint sequences for each drone to accomplish its tasks using the algorithms proposed in Section 6.5. The generated waypoint sequences are sent to the data manager, which forwards them to the corresponding drone controllers. The whole drone data collection and flight planning process are depicted in Fig. 6.15.

**Visualizing fire setting status.** DOME provides a comprehensive **dashboard** as shown in Figure 6.14, that enables users to visualize critical aspects of wildland fires, including current fire locations, predicted fire spread, drone positions, wind conditions, and air quality. We developed an HTML-based website that retrieves this information from the edge server through HTTP requests and updates the data periodically. The data manager responds to these requests by querying data from the database and sending it to the dashboard for

display. During monitoring process, all received images and fire detection results will be shown in our dashboard. The dashboard will display recently received photos with detected bounded fires. Using the drone location and data capture time, we will create fire detection events that provide the location of the fires with granular detail down to the grid cells. The detected events will trigger updates of the fire status in the dashboard.

## 6.8 Summary and Discussion

In this chapter, we design a DOME system 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.

# Chapter 7

## Conclusion

This chapter summarizes the primary contributions of this thesis, which focus on the exploration of integrating mobility into diverse time-sensitive IoT applications. Throughout this research, we have delved into the challenges and opportunities presented by this integration, aiming to enhance the sensing and networking capabilities of IoT systems. In the following sections, we will elaborate on the valuable lessons gained from our work and outline the potential directions for future research.

### 7.1 Summary of Thesis Contributions

This thesis explores the integration of mobile entities with planned mobility patterns into time-sensitive IoT systems to enhance their sensing and networking capabilities. Planned mobility includes both predefined mobility, where mobile entities, such as buses, follow fixed trajectories, and on-demand mobility, where entities like drones or robots respond to specific user-initiated tasks. In three diverse scenarios, we investigate the utilization of mobile devices in IoT applications, ranging from urban to remote areas, each presenting unique challenges

and trade-offs during implementation.

The first scenario focuses on smart city applications in urban settings, utilizing pervasive media-rich sensor networks with diverse data transmission delays and priority. To address the limitation of local network resources for long-range data transmission, we propose using the mobility of public transit fleets (buses) and network infrastructures on bus routes to establish a backbone network for cost-effective sensor data transmission. We develop approaches for network infrastructure deployment and data collection planning, considering the heterogeneity of delay tolerance and priority of sensor data, as well as the trade-off between data delivery delay/loss and network infrastructure installation cost. The proposed approaches are evaluated using a real-world bus network in Orange County, CA, and their applicability and efficiency are studied in comparison to several other approaches. Furthermore, the demonstrated data collection approach can be seamlessly extended to include other entities like railways and city fleets by merely providing their trajectories and schedules.

In the second scenario, we explore the use of drones to enhance sensing coverage in mission-critical IoT applications, particularly in high-rise fire monitoring. In this context, accessing in-situ sensors is challenging due to factors such as power shortages, damage, and various other obstacles. We design and implement a drone-based IoT platform for real-time data collection in fire settings. The system includes visualization of monitoring areas based on data analysis results, user interfaces that provide commands to drones for monitoring tasks, and automatic flight planning for continuous monitoring. To optimize data collection, we propose multiple-drone flight planning approaches, considering the heterogeneity of monitoring tasks in terms of periods and priorities, as well as the trade-off between sensing coverage and data quality. The proposed algorithms are evaluated using a simulated high-rise fire scenario with real building structures in UCI. Additionally, we assess the applicability of our system by implementing its prototype in a lab-based testbed, simulating mockup high-rise fires.



In the third scenario, we focus on utilizing drones to assist mobile sensing and data transmission in remote areas with limited in-situ sensors and poor network conditions, specifically for wildland fire monitoring. We leverage the CPS control loop to automate the drone-based monitoring system by enabling real-time perception of the physical world based on sensor data, automatic task generation for mobile entities, and dynamic planning and control of their movements to continuously monitor dynamic environments. We propose a rule-based task generation procedure for spatio-temporal monitoring requirements based on fire status and prediction. We investigate approaches for multiple-drone flight planning, considering data collection timeliness, the trade-off between sensing coverage and data quality, and network disconnection during flights. These proposed approaches are thoroughly evaluated by comparing them with baseline algorithms in simulated wildland fire burns at the Blodgett Forest Research Station. Additionally, we implement the proposed drone-based monitoring system in a lab-based testbed with mockup wildland fires to assess its applicability and performance in realistic conditions.

In the latter two scenarios, this thesis introduces a novel drone-assisted monitoring system tailored to monitor dynamic environments, with high-rise and wildland fires as case studies. A synthesis of approaches from multiple fields and disciplines is required to achieve this system. The system's design entails creating semantic representations of user-specific high-level monitoring needs using missions and tasks, taking inspiration from real-time systems terminology. These designs account for spatial sensor data collection requirements and temporal necessities for data transmission to enable continuous monitoring in dynamic environments. Furthermore, rule-based approaches are incorporated using logical frameworks, automating task generation and minimizing human intervention. In planning and scheduling mobile entity actions, operations research techniques are harnessed to optimize the deployment and movement of entities, aligning with sensor coverage needs and diverse monitoring objectives' temporal demands. This approach encompasses heuristic methods grounded in domain-specific knowledge and application needs alongside metaheuristic techniques like lo-

cal search approaches in high-rise fire scenarios. Furthermore, the thesis incorporates AI concepts such as Markov decision processes and heuristic search methodologies to enhance action planning and scheduling, especially relevant to wildland fire scenes. The thesis also addresses challenges such as network disconnections during entity movement and 3D obstacle avoidance, utilizing path planning methods derived from robotics, specifically in high-rise fire scenarios. In the system implementation phase, we delved into network protocols and message exchange models among diverse components. This enabled sensor data collection, data processing, and its presentation on the dashboard, along with an interface for user instructions input. Additionally, we implemented a control procedure to guide the movement of entities. Our proposed drone-based monitoring system can be readily extended to different scenarios through the customization of missions and task generation rules informed by domain knowledge. Additionally, the system accommodates diverse sensor configurations to implement our flight planning algorithms while also addressing challenges related to network disconnectivity conditions. This flexibility and adaptability collectively enable the generalization of our system.

## 7.2 Key Observations and Insights

Through our exploration of integrating mobility into diverse IoT applications, we have learned the following valuable lessons.

**Preliminary and domain knowledge are critical for designing IoT systems.** When planning the deployment of monitoring entities in IoT systems, considering both preliminary knowledge and domain expertise is critical and helpful. In Chapter 4, for urban scenarios, the awareness of in-situ sensor placement and the schedules of public transit fleets are crucial for planning data transmission using mobile vehicles. In Chapter 5, when deploying drones for mobile sensing in high-rise fire settings, understanding building structures and layouts

aids in predicting fire evolution and identifying critical areas for monitoring. Similarly, in the wildland fire settings of Chapter 6, knowledge about forest topology and fire ignition plans helps in planning drone flights to monitor areas at risk of fire arrival. Additionally, our IoT platform for monitoring fire settings is designed based on domain knowledge from fire experts. We determine monitoring objectives in high-rise fire scenarios based on firefighters' suggestions, focusing on factors such as building ventilation and window openings, which significantly impact fire spread. For wildland fires, we employ physical fire models for predictions based on perceived current fire status. Our design also includes rules, guided by firefighters' needs, to generate monitoring requirements that dictate drones' motion. For example, in prescribed fire settings, we task drones to inspect fire intensity and flame length to facilitate the controlled burning of vegetation and track the fire front according to fire predictions for quickly detecting potential fire spots and escape routes.

**Balance data transfer timing and network infrastructure cost during mobile entity-assisted data transmission.** In this thesis (Chapter 4), we explored the tradeoff between the timing of sensor data collection and the cost of network infrastructure installation. Our study involves mobile entities that collect data from local sensors and upload them to access points along their routes, enabling cost-effective long-range data transmission. When deploying network access points along the routes of mobile entities, we encounter two conflicting factors. Increasing the number of access points can reduce data transmission delay, as entities encounter access points more frequently and promptly upload data after collection. However, this leads to higher installation costs. Conversely, insufficient access points along the routes can result in delayed data uploading and potential data loss. To strike a balance between data transmission delay and installation costs, we consider the specific delay tolerance of diverse types of sensor data. By prioritizing access points near data with lower delay tolerance, we optimize the deployment to minimize data transfer delay and data loss while adhering to the constraint of installation cost.

**Tradeoffs between sensing coverage and data quality in mobile sensing.** In our exploration of the placement of aerial mobile entities (drones) for capturing sensor data (in Chapters 5 and 6), we encounter another tradeoff. The location of drones (and the sensors mounted on them) significantly impacts the quality of sensor data, which directly relates to the level of detail in the observations of targets. Visual sensors like RGB and thermal cameras offer higher data quality when they are positioned closer to the observed area. This proximity allows for more detailed observations and higher data resolution. However, this comes at the cost of smaller sensor coverage, as the drones can only observe a limited area from close range. Conversely, positioning the drones further from the targets results in larger sensor coverage but lower data quality. To balance sensor coverage and data quality, we first determine the required data quality (spatial resolution) for detecting specific events, such as fires or humans. By establishing a mapping between data quality and event detection accuracy, we can quantify the value of data quality for monitoring specific objectives. Additionally, we assess the sensor coverage range concerning the diverse distances to the monitoring objectives to ensure specific targeting areas can be covered when the drone is at a particular location. When planning drones' movements to monitor specific areas, we prioritize sensor coverage over data quality. This decision is guided by the intuition that within limited data collection timeframes, it is crucial to observe monitoring targets as much as possible, even if it means accepting data of acceptable quality initially. If there is additional time available, we can then focus on enhancing the data quality. This strategy effectively handles the tradeoff between sensor coverage and data quality while efficiently monitoring critical areas.

**Formulating temporal requirements in continuous monitoring with decay functions or deadlines.** During the development of flight planning approaches for monitoring dynamic environments with drones (in Chapters 5 and 6), we faced the challenge of formulating continuous monitoring requirements to guide the drones' actions. Each drone action involves capturing sensor data to cover specific sub-areas and detect particular targets. However, due to the dynamic nature of the monitoring areas, such as in fire settings, and

the limitations of mobile entities to provide constant coverage of the entire areas, we must guide drones to repetitively monitor specific areas in accordance with the time-sensitivity for data collection specified by domain experts. To address the continuous data collection requirements with specific time sensitivity, we explored two approaches for formulating the temporal requirements for drone data collection. In Chapter 5, we formulated the monitoring requirements of a specific area using a decay function based on the time-sensitive delay tolerance of sensor data collection. Each sub-area requiring monitoring maintains its information accuracy to track the historical collected data perceived at that location, and this accuracy function decays over time as the gap between consecutive observations increases. The drone’s monitoring process aims to maximize the information accuracy of all monitoring sub-areas. While this formulation accurately tracks the dynamics of each sub-area, it introduces complexity by modeling information accuracy as a continuous function, which makes it challenging to track task accomplishment at each delay tolerance period. In Chapter 6, we adopted a simpler approach by modeling the monitoring requirements of each monitoring objective as periodic tasks, each with a start time and a deadline. This time window corresponds to the time-sensitive period required for data collection and observation. The drone is tasked with observing the designated objectives within the specified time window; failure to do so results in missing the subtasks. The goal of this formulation is to maximize the completeness of all subtasks, providing an easier way to track task accomplishment at each period.

**Enhancing autonomy in IoT systems through CPS control loop.** Automation is crucial when designing mobile entities-based IoT systems, especially for monitoring large-scale areas such as wildland fires. In such vast and remote areas, human ground-based monitoring of mobile entities becomes challenging or impossible due to limited visibility. Additionally, in forested areas, there can be a loss of network connectivity with aerial vehicles. To achieve system autonomy, we implement multiple components following the CPS control loop. The first component is the data analysis module, which abstracts meaningful information about

the physical world from sensor data. In addition, we design system semantics to represent the status of the monitored environment based on data analysis results. For instance, in the context of fire monitoring, we partition the entire monitoring area into smaller cells using a grid system. This approach enables us to track the burning status at each cell, allowing for the effective localization of fires. Furthermore, we create a rule-based task-generation procedure that can automatically generate monitoring requirements for drones. Here, we utilize if-then rules specific to fire experts' knowledge to generate spatial and temporal data collection requirements for drones based on multiple predefined conditions related to the current perceived fire status and prediction of future fire evolution. Given the generated tasks, we also designed the flight planning component to automatically plan and schedule drone flights to fulfill these tasks.

By enabling the perception of the physical world's status and rule-based task generation, we can automatically guide drones to collect sensor data for user-specific purposes. This framework can also be extended to other scenarios by modifying the data analysis procedures and rules based on specific domain knowledge.

## 7.3 Future Work

Building upon the insights gained from our current study, we can explore several promising areas that have the potential to significantly enhance the utilization of mobile entities in IoT systems.

- **Onboard computing and flight adaptation.** In our current study, we conduct periodic planning of aerial vehicles capturing sensor data for monitoring dynamic environments. However, this periodic planning approach may not be flexible enough to react to unexpected events that occur during the planned duration. To enable dynamic

monitoring adjustments based on unexpected events during flight, it is essential to investigate the implementation of onboard computing capabilities in drones. This would empower the drones to detect events based on onboard data analysis, such as using temperature sensors to detect fires in proximity, and automatically adapt their actions for safety or other concerns in real-time. For instance, if a drone detects a fire alarm during its flight, it can dynamically adjust its trajectory to respond to the emergency. Similarly, in case of a sudden network disconnection, the drone can adapt its flight to upload data promptly when the connection is restored. Incorporating onboard computing can enhance the responsiveness and efficiency of mobile entities in IoT systems, allowing them to address unexpected situations more effectively.

- **Utilizing formal methods for verifying system properties.** Exploring the verification of system robustness and safety of mobile entities using formal methods-based approaches is another important area for future work. By applying formal methods-based approaches, we can systematically analyze and validate the system's behavior, identify potential vulnerabilities or safety concerns, and ensure that the mobile entities operate efficiently and safely in various scenarios. First, using formal methods approach, we can verify the correctness of the system's operation. For example, we can assess whether tasks can be completed within specific time durations based on the rule-based task generation procedure. Additionally, we can verify if data can be transmitted from drones to edge servers before the deadline of data collection. Moreover, exploring the verification of system robustness can involve assessing the mobile entities' resilience to various environmental factors and external disturbances. For example, in the case of drones deployed for monitoring in extreme weather conditions or challenging terrains, it is essential to verify their ability to maintain safe distances from fires, avoid potential collisions with other mobile entities, and respond appropriately to unexpected events, such as loss of power during task execution. Furthermore, formal methods can be applied to evaluate the system's adaptability and scalability.

For example, it becomes crucial to ensure that the mobile entities (drones) can effectively handle the updating of monitoring tasks or the addition/removal of drones, and dynamically adjust their operations to changing demands.

- **Exploring multi-Hop networks with mobile entities for enhanced monitoring processes.** In this thesis, our primary focus was on one-hop data transmission between mobile entities and the edge server. However, we recognize the potential of exploring network communication among mobile entities, with an emphasis on creating multi-hop data transmission among drones. This approach can effectively complement the limited data transmission range and save drones' flying time for uploading data. Given the fast and flexible movement of aerial vehicles, our research will delve into investigating the optimal placement of drones to establish a robust mesh network. This will involve determining the roles of individual drones, such as access points, data relayers, or sensor data collectors, within this mesh network. Addressing the issue of connection failures between drones becomes crucial in this context. Further research should concentrate on developing efficient algorithms and protocols for rerouting data transmission routes, ensuring data continuity and recovery in case of network disruptions. By exploring multi-hop data transmission and mesh network establishment, we can enhance the data transmission efficiency and overall performance of mobile entities in IoT systems. This investigation will contribute to a more resilient and well-connected network of mobile entities, capable of seamless collaboration in real-time data collection and monitoring tasks.
- **Integrating mixed mode mobility into IoT systems.** In this thesis, we focused on exploring scenarios each of which involves a single type of mobile entities. In our future work, we plan to investigate the incorporation of mixed mode mobility, which includes both ground and aerial vehicles, in our IoT system. By combining these two types of mobile entities, we can leverage their unique strengths to address various



challenges more effectively. Aerial vehicles excel in conducting sensing and actuation in challenging environments due to their quick deployment and agile movement capabilities. On the other hand, ground vehicles offer advantages such as larger payload capacity and suitability for acting as power suppliers, overcoming the limitations of aerial vehicles in terms of energy and loading weight. The collaboration of ground and aerial vehicles to fulfill specific monitoring tasks presents several challenges, including role assignment, resource allocation, and motion planning. To facilitate efficient cooperation, we categorize the devices based on their functionalities, including sensors responsible for data collection, actuators responsible for performing various actions, decision makers handling computing and analysis, and auxiliary facilities such as access points for networking assistance or energy suppliers. Once the roles are assigned, we need to plan the actions of mobile devices to fulfill specific service tasks, framing the problem as a cooperative multi-agent planning challenge. This approach allows us to optimize the coordination and cooperation among the mobile entities, enabling them to work together seamlessly and achieve higher efficiency in data collection and mobile sensing tasks.

- **Exploring edge computing through runtime resource allocation and adaptation.** In our current IoT system, the data analysis component relies on centralized data processing at the edge server. However, for future work, we aim to explore distributed edge computing, which involves allocating data processing and analysis tasks among multiple edge servers or onboard computing components. By adopting a distributed approach, we can reduce the computing time required for data processing and analysis, leading to improved efficiency and faster response times. The exploration of distributed edge computing and resource allocation involves several research challenges. One of the main problems is optimizing the workload allocation among multiple edge servers and onboard components, considering factors such as running time, data transmission delay and power consumption. This requires developing effi-

cient algorithms and techniques to determine the best distribution of data processing tasks across the available resources to minimize overall processing time and reduce communication overhead. Another critical aspect is the dynamic adaptation of data processing task to respond to the dynamic status of the edge computing servers. For instance, in the case of certain servers running out of power or experiencing failures, the system needs to quickly redistribute the workload to other available servers to ensure continuous and uninterrupted data analysis.

# Bibliography

- [1] Digitanimal. <https://digitanimal.com/>, 2018.
- [2] E-Pasto. <https://www.epasto.fr>, 2018.
- [3] Rating Camera Performance, DRI & DORI Explained. <https://tinyurl.com/3wex9uzu>, 2021.
- [4] PyKnow: Expert Systems for Python. <https://github.com/buguroo/pyknow>, 2022.
- [5] DOME. <https://github.com/Fangqierin/DOME-Monitoring-System>, 2023.
- [6] U. Acer, P. Giaccone, D. Hay, G. Neglia, and S. Tarapiah. Timely Data Delivery in a Realistic Bus Network. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 446–450, Apr. 2011.
- [7] C. Aeronautiques et al. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.
- [8] M. A. Akhloufi, A. Couturier, and N. Castro. Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. *Drones*, 2021.
- [9] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392, 2014.
- [10] E. Alba and R. Martí. *Metaheuristic procedures for training neural networks*, volume 35. Springer Science & Business Media, 2006.
- [11] M. Aljehani and M. Inoue. Safe map generation after a disaster, assisted by an unmanned aerial vehicle tracking system. *IEEJ Transactions on Electrical and Electronic Engineering*, 14(2):271–282, 2019.
- [12] S. H. Alsamhi, O. Ma, M. S. Ansari, and F. A. Almalki. Survey on collaborative smart drones and internet of things for improving smartness of smart cities. *Ieee Access*, 7:128125–128152, 2019.
- [13] S. H. Alsamhi, O. Ma, M. S. Ansari, and S. K. Gupta. Collaboration of drone and internet of public safety things in smart cities: An overview of qos and network performance optimization. *Drones*, 3(1):13, 2019.

- [14] S. Amador, S. Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.
- [15] Apache Software Foundation. Hadoop.
- [16] J. Arshad, A. U. Rehman, M. T. B. Othman, M. Ahmad, H. B. Tariq, M. A. Khalid, M. A. R. Moosa, M. Shafiq, and H. Hamam. Deployment of wireless sensor network and iot platform to implement an intelligent animal monitoring system. *Sustainability*, 14(10):6249, 2022.
- [17] O. Avatefipour and F. Sadry. Traffic management system using iot technology-a comparative review. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 1041–1047. IEEE, 2018.
- [18] E. D. Ayele, N. Meratnia, and P. J. Havinga. Towards a new opportunistic iot network architecture for wildlife monitoring system. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, pages 1–5. IEEE, 2018.
- [19] H. Azpúrua, G. M. Freitas, and e. Macharet, D. G. Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys. *Robotica*, 36(8), 2018.
- [20] M. Babar and F. Arif. Real-time data processing scheme using big data analytics in internet of things based smart transportation environment. *Journal of Ambient Intelligence and Humanized Computing*, 10:4167–4177, 2019.
- [21] J. A. Baijnath-Rodino and e. a. Li, S. Historical seasonal changes in prescribed burn windows in california. *Science of the total environment*, 2022.
- [22] J. Ballesteros, B. Carbunar, M. Rahman, N. Rishe, and S. Iyengar. Towards safe cities: A mobile and social networking approach. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2451–2462, 2013.
- [23] R. W. Beard and T. W. M. et al. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18, 2002.
- [24] Beatrice Rasciute. What Is IoT? All You Need to Know About the Internet of Things, November 2021. <https://www.ipxo.com/blog/what-is-iot/>,.
- [25] B. Behsaz, M. R. Salavatipour, and Z. Svitkina. New Approximation Algorithms for the Unsplittable Capacitated Facility Location Problem. In *Algorithm Theory - Scandinavian Symposium and Workshops*, pages 237–248, Jun. 2016.
- [26] J. Bellingham et al. Receding horizon control of autonomous aerial vehicles. In *Proceedings of the 2002 American Control Conference*, volume 5, 2002.
- [27] L. Bengtsson, X. Lu, A. Thorson, R. Garfield, and J. Von Schreeb. Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: a post-earthquake geospatial study in haiti. *PLoS medicine*, 8(8):e1001083, 2011.

- [28] F. Z. Benhamida, A. Bouabdellah, and Y. Challal. Using delay tolerant network for the internet of things: Opportunities and challenges. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 252–257. IEEE, 2017.
- [29] K. Benson et al. Scale: Safe community awareness and alerting leveraging the internet of things. *IEEE Communications Magazine*, 53(12), 2015.
- [30] T. Black, V. Mak, P. Pathirana, and S. Nahavandi. Using Autonomous Mobile Agents for Efficient Data Collection in Sensor Networks. In *World Automation Congress (WAC)*, Aug. 2006.
- [31] S. Bonadies and S. A. Gadsden. An overview of autonomous crop row navigation strategies for unmanned ground vehicles. *Engineering in Agriculture, Environment and Food*, 12(1):24–31, 2019.
- [32] O. Bräysy and M. Gendreau. Tabu search heuristics for the vehicle routing problem with time windows. *Top*, 10(2):211–237, 2002.
- [33] D. Buezas. *Constraint-Based Modeling of Minimum Set Covering: Application to Species Differentiation*. PhD thesis, Faculdade de Ciências e Tecnologia, 2010.
- [34] T. Buratowski, J. Garus, M. Giergiel, and A. Kudriashov. Real-time 3d mapping in isolated industrial terrain with use of mobile robotic vehicle. *Electronics*, 11(13):2086, 2022.
- [35] M. N. Bygi. 3D Visibility Graph. *Computer Engr.*, 2007.
- [36] T. M. Cabreira, L. B. Brisolara, and P. R. Jr. Ferreira. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 2019.
- [37] F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE transactions on intelligent transportation systems*, 12(1):141–151, 2010.
- [38] P. Cao, Z. Fan, R. X. Gao, and J. Tang. Solving Configuration Optimization Problem with Multiple Hard Constraints: An Enhanced Multi-Objective Simulated Annealing Approach. *arXiv preprint arXiv:1706.03141*, (860), Jun. 2017.
- [39] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE communications surveys & tutorials*, 21(3):2419–2465, 2019.
- [40] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations research*, 47, 1999.
- [41] A. Cardoso, J. Pereira, L. Nóbrega, P. Gonçalves, P. Pedreiras, and V. Silva. Sheepit: Activity and location monitoring. In *Proceedings of the INForum*, pages 1–12, 2018.

- [42] A. A. Chaudhry, R. Mumtaz, S. M. H. Zaidi, M. A. Tahir, and S. H. M. School. Internet of things (iot) and machine learning (ml) enabled livestock monitoring. In *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, pages 151–155. IEEE, 2020.
- [43] C. Chen, D. Zhang, Z.-H. Zhou, N. Li, T. Atmaca, and S. Li. B-planner: Night bus route planning using large-scale taxi gps traces. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 225–233, 2013.
- [44] S. Chen, J. Dong, P. Ha, Y. Li, and S. Labi. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, 36(7):838–857, 2021.
- [45] 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*, 40(5):1259–1272, May 2017.
- [46] H. Cheng and G. V. Hadjisophocleous. Dynamic modeling of fire spread in building. *Fire Safety Journal*, 46:211–224, 2011.
- [47] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. *Intl. Conf. on Intelligent Agent Technology*, 2004.
- [48] N. Chiaraviglio and e. Artés, T. Automatic fire perimeter determination using modis hotspots information. In *12th Int. Conf. on e-Science*, 2016.
- [49] W. Choi, J. Kim, S. Lee, and E. Park. Smart home and internet of things: A bibliometric study. *Journal of Cleaner Production*, 301:126908, 2021.
- [50] H. Choset. Coverage for robotics—a survey of recent results. *Annals of mathematics and artificial intelligence*, 31:113–126, 2001.
- [51] S. Chu, P. Wei, X. Zhong, X. Wang, and Y. Zhou. Deployment of a Connected Reinforced Backbone Network with a Limited Number of Backbone Nodes. *IEEE Transactions on Mobile Computing*, 12(6):1188–1200, Apr. 2013.
- [52] I. I. Cplex. V12. 1: User’s manual for cplex. *Intel.*, 2009.
- [53] A. Crooks, A. Croitoru, A. Stefanidis, and J. Radzikowski. # earthquake: Twitter as a distributed sensor system. *Transactions in GIS*, 17(1):124–147, 2013.
- [54] M. Crosby et al. Automated agent decomposition for classical planning. In *Intel. Conference on Automated Planning and Scheduling*, 2013.
- [55] P. H. Cruz Caminha, F. Ferreira da Silva, R. Gonçalves Pacheco, R. de Souza Couto, P. Braconnot Velloso, M. E. Mitre Campista, and L. H. M. K. Maciel Kosmalski Costa. Sensingbus: Using bus lines and fog computing for smart sensing the city. *IEEE Cloud Computing*, 5(5):58–69, 2018.

- [56] F. Cui. Deployment and integration of smart sensors with iot devices detecting fire disasters in huge forest environment. *Computer Communications*, 150:818–827, 2020.
- [57] C. D. F. and o. Giorgio. Coverage path planning for uavs photogrammetry with energy and resolution constraints. *Journal of Intel. & Robotic Systems*, 83(3), 2016.
- [58] S. M. S. M. Daud, M. Y. P. M. Yusof, C. C. Heo, L. S. Khoo, M. K. C. Singh, M. S. Mahmood, and H. Nawawi. Applications of drone in disaster management: A scoping review. *Science & Justice*, 62(1):30–42, 2022.
- [59] B. De Longueville, R. S. Smith, and G. Luraschi. ” omg, from here, i can see the flames!” a use case of mining location based social networks to acquire spatio-temporal data on forest fires. In *Proceedings of the 2009 international workshop on location based social networks*, pages 73–80, 2009.
- [60] V. R. Desaraju et al. Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In *IEEE ICRA*, 2011.
- [61] DJI. Mobile SDK. <https://developer.dji.com/mobile-sdk/>.
- [62] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, and S. Behnke. Multi-layered mapping and navigation for autonomous micro aerial vehicles. *Journal of Field Robotics*, 33(4):451–475, 2016.
- [63] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Trans. Sen. Netw.*, 6(1), jan 2010.
- [64] E. et al. UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. *Intl. Conf. on Unmanned Aircraft Systems*, pages 274–283, 2014.
- [65] M. et al. *Fire fighting tactics under wind driven conditions: laboratory experiments*. Fire Protection Research Foundation, 2009.
- [66] Y. Eun et al. Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. *Journal of aircraft*, 2009.
- [67] T.-Y. Fan, F. Liu, J.-W. Fang, N. Venkatasubramanian, and C.-H. Hsu. Enhancing situational awareness with adaptive firefighting drones: Leveraging diverse media types and classifiers. In *Proc. of the 13th ACM Multimedia Systems Conference*, Athlone, Ireland, 2022.
- [68] M. A. Finney. *FARSITE, Fire Area Simulator—model development and evaluation*. Number 4. US Department of Agriculture, Forest Service, 1998.
- [69] FireRescue1. 5 Drone Technologies for Firefighting, Mar. 2014. <https://tinyurl.com/y41dl8mf>, Last accessed on 2019-11-10.

- [70] P. M. e. a. França. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29(3):267–275, 1995.
- [71] E. Fresk, K. Ödmark, and G. Nikolakopoulos. Ultra wideband enabled inertial odometry for generic localization. *IFAC-PapersOnLine*, 50(1):11465–11472, 2017.
- [72] J. Gai, L. Xiang, and L. Tang. Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle. *Computers and Electronics in Agriculture*, 188:106301, 2021.
- [73] Y. Gao, W. Dong, K. Guo, X. Liu, Y. Chen, X. Liu, J. Bu, and C. Chen. Mosaic: A low-cost mobile sensing system for urban air quality monitoring. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.
- [74] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada. The evolution of robotics research. *IEEE Robotics & Automation Magazine*, 14(1):90–103, 2007.
- [75] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.
- [76] A. Gatouillat, Y. Badr, B. Massot, and E. Sejdić. Internet of medical things: A review of recent contributions dealing with cyber-physical systems in medicine. *IEEE internet of things journal*, 5(5):3810–3822, 2018.
- [77] K. A. Ghamry and Y. Zhang. Cooperative control of multiple uavs for forest fire monitoring and detection. In *MESA*, 2016.
- [78] A. A. Ghapar, S. Yussof, and A. A. Bakar. Internet of things (iot) architecture for flood data management. *International journal of future generation communication and networking*, 11(1):55–62, 2018.
- [79] A. Goudarzi, F. Ghayoor, M. Waseem, S. Fahad, and I. Traore. A survey on iot-enabled smart grids: Emerging, applications, challenges, and outlook. *Energies*, 15(19):6984, 2022.
- [80] R. Grosso, U. Mecca, G. Moglia, F. Prizzon, and M. Rebaudengo. Collecting built environment information using uavs: Time and applicability in building inspection activities. *Sustainability*, 12(11):4731, 2020.
- [81] A. Gunawan and e. H. C. Lau. Orienteering problem: A survey of recent variants, solution approaches and applications. *EJOR*, 2016.
- [82] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou. Opportunistic iot: Exploring the harmonious interaction between human and the internet of things. *Journal of Network and Computer Applications*, 36(6):1531–1539, 2013.
- [83] S. Guo, M. Derakhshani, M. Falaki, U. Ismail, R. Luk, E. Oliver, S. U. Rahman, A. Seth, M. Zaharia, and S. Keshav. Design and implementation of the kiosknet system. *Computer Networks*, 55(1):264–281, 2011.



- [84] W. He, G. Yan, and L. D. Xu. Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595, 2014.
- [85] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming*, 1, 1971.
- [86] S. Hodges, S. Taylor, N. Villar, J. Scott, D. Bial, and P. T. Fischer. Prototyping connected devices for the internet of things. *Computer*, 46(2):26–34, 2012.
- [87] J. Hoffmann et al. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, 2001.
- [88] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [89] T. Huang, J. Li, S. Koenig, and B. Dilkina. Anytime multi-agent path finding via machine learning-guided large neighborhood search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9368–9376, 2022.
- [90] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, 2006.
- [91] 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*, 10(1):23–29, 2018.
- [92] N. Indra Er, K. Deep Singh, J.-M. Bonnin, N. E. Indra, and K. Deep SINGH. On the Performance of VDTN Routing Protocols with V2X Communications for Data Delivery in Smart Cities. In *International Workshop on System Safety & Security (IWSSS)*, pages 1–2, Aug. 2017.
- [93] A. Jadon et al. Firenet: a specialized lightweight fire & smoke detection model for real-time iot applications. *arXiv preprint arXiv:1905.11922*, 2019.
- [94] S. I. Jiménez-Jiménez, W. Ojeda-Bustamante, R. E. Ontiveros-Capurata, and M. d. J. Marcial-Pablo. Rapid urban flood damage assessment using high resolution remote sensing data and an object-based approach. *Geomatics, Natural Hazards and Risk*, 11(1):906–927, 2020.
- [95] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, 2002.

- [96] A. Juels, R. L. Rivest, and M. Szydlo. The blocker tag: Selective blocking of rfid tags for consumer privacy. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 103–111, 2003.
- [97] G. Kahn, P. Abbeel, and S. Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021.
- [98] A. Kamilaris, F. Gao, F. X. Prenafeta-Boldu, and M. I. Ali. Agri-iot: A semantic framework for internet of things-enabled smart farming applications. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 442–447. IEEE, 2016.
- [99] L. Kang. A Public Transport Bus as a Flexible Mobile Smart Environment Sensing Platform for IoT. In *International Conference on Intelligent Environments (IE)*, pages 1–8, Sep. 2016.
- [100] L. Kang, S. Poslad, W. Wang, X. Li, Y. Zhang, and C. Wang. A Public Transport Bus as a Flexible Mobile Smart Environment Sensing Platform for IoT. In *International Conference on Intelligent Environments (IE)*, pages 1–8, Sep. 2016.
- [101] L. Kang, S. Poslad, W. Wang, X. Li, Y. Zhang, and C. Wang. A public transport bus as a flexible mobile smart environment sensing platform for iot. In *2016 12th International Conference on Intelligent Environments (IE)*, pages 1–8. IEEE, 2016.
- [102] K. Kanistras, G. Martins, M. J. Rutherford, and K. P. Valavanis. A survey of unmanned aerial vehicles (uavs) for traffic monitoring. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 221–234. IEEE, 2013.
- [103] D. Kazakov and D. Kudenko. Machine learning and inductive logic programming for multi-agent systems. In *ECCAI Advanced Course on Artificial Intelligence*, pages 246–270. Springer, 2001.
- [104] H. Kellerer, U. Pferschy, and D. Pisinger. Introduction to NP-Completeness of Knapsack Problems. In *Knapsack Problems*, pages 483–493, 2004.
- [105] F. Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378, 2012.
- [106] A. Ker and K. Teemu. Simulating Mobility and DTNs with the ONE. *Journal of Communications*, 5(2):92–105, Sep. 2010.
- [107] A. Keränen. “Opportunistic Network Environment Simulator”, 2008.
- [108] A. W. Khan, A. H. Abdullah, M. H. Anisi, and J. I. Bangash. A Comprehensive Study of Data Collection Schemes Using Mobile Sinks in Wireless Sensor Networks. *Sensors*, pages 2510–2548, May 2014.
- [109] A. Khanna and S. Kaur. Internet of things (iot), applications and challenges: a comprehensive review. *Wireless Personal Communications*, 114:1687–1762, 2020.

- [110] M. S. Kiarostami, M. R. Daneshvaramoli, S. K. Monfared, D. Rahmati, and S. Gorgin. Multi-agent non-overlapping pathfinding with monte-carlo tree search. In *2019 IEEE Conference on Games (CoG)*, pages 1–4. IEEE, 2019.
- [111] D. Kim, R. N. Uma, B. H. Abay, W. Wu, W. Wang, and A. O. Tokuta. Minimum Latency Multiple Data Mule Trajectory Planning in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 13(4), May 2014.
- [112] H. Kim, L. Mokdad, and J. Ben-Othman. Designing uav surveillance frameworks for smart city and extensive ocean with differential perspectives. *IEEE Communications Magazine*, 56(4):98–104, 2018.
- [113] J. Kim and H. I. Son. A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. *IEEE Access*, 8, 2020.
- [114] Y. Kim, D. W. Gu, and I. Postlethwaite. Real-time optimal mission scheduling and flight path selection. *IEEE Transactions on Automatic Control*, 52(6):1119–1123, 2007.
- [115] E. King et al. Coordination and control experiments on a multi-vehicle testbed. In *Proceedings of the 2004 American Control Conference*, volume 6, 2004.
- [116] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., USA, 2005.
- [117] C. Konstantopoulos, G. Pantziou, and D. Gavalas. A Rendezvous-Based Approach Enabling Energy-Efficient Sensory Data Collection with Mobile Sinks. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 23(5):809–817, Sep. 2011.
- [118] H. Korala, D. Georgakopoulos, P. P. Jayaraman, and A. Yavari. A survey of techniques for fulfilling the time-bound requirements of time-sensitive iot applications. *ACM Comput. Surv.*, 54, sep 2022.
- [119] S. Koulali, E. Sabir, T. Taleb, and M. Azizi. A green strategic activity scheduling for uav networks: A sub-modular game perspective. *IEEE Communications Magazine*, 54(5):58–64, 2016.
- [120] P. Kuila and P. K. Jana. *“Clustering and Routing Algorithms for Wireless Sensor Networks: Energy Efficiency Approaches”*. Chapman and Hall/CRC, Sep. 2017.
- [121] R. Kulkarni and P. R. Bhave. Integer programming formulations of vehicle routing problems. *European journal of operational research*, 20(1):58–67, 1985.
- [122] 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.
- [123] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2013.

- [124] J. Lanza, L. Sánchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, and V. Gutiérrez. Large-scale mobile sensing enabled internet-of-things testbed for smart city services. *International Journal of Distributed Sensor Networks*, 11(8):785061, 2015.
- [125] G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In *The vehicle routing problem*, pages 109–128. SIAM, 2002.
- [126] A. Lavric, A. I. Petrariu, and V. Popa. Sigfox communication protocol: The new era of iot? In *2019 international conference on sensing and instrumentation in IoT Era (ISSI)*, pages 1–4. IEEE, 2019.
- [127] S. Leary, M. Deittert, and J. Bookless. Constrained UAV mission planning: A comparison of approaches. *Proceedings of the IEEE Intl. Conf. on Computer Vision*, pages 2002–2009, 2011.
- [128] S. K. Lee, M. Bae, and H. Kim. Future of iot networks: A survey. *Applied Sciences*, 7(10):1072, 2017.
- [129] Leonardo DRS, Electro-Optical & Infrared Systems. How To Assess Thermal Camera Range For Site Design, White Paper. [shorturl.at/osyHT](http://shorturl.at/osyHT).
- [130] V. Lesch, M. Züfle, A. Bauer, L. Iffländer, C. Krupitzer, and S. Kounev. A literature review of iot and cps—what they are, and what they are not. *Journal of Systems and Software*, 200:111631, 2023.
- [131] L. Li, Y. Liu, Z. Li, and L. Sun. R2R: Data Forwarding in Large-Scale Bus-Based Delay Tolerant Sensor Networks. In *IET International Conference on Wireless Sensor Network (IET-WSN)*, pages 27 – 31, Nov. 2010.
- [132] C. M. D. A. Lima, E. A. D. Silva, and P. B. Velloso. Performance evaluation of 802.11 iot devices for data collection in the forest with drones. In *Global Comms. Conf.*, 2018.
- [133] C. R. Lin and M. Gerla. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, Sep. 1997.
- [134] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. In *Intl. Conf. on Decision and Control*, volume 2, 2003.
- [135] F. Liu and e. a. T. Y. Fan. Dragonfly: Drone-assisted high-rise monitoring for fire safety. In *Int. Symposium on Reliable Distributed Systems*, 2021.
- [136] X. Liu, T. Yang, and B. Yan. Internet of things for wildlife monitoring. In *2015 IEEE/CIC International Conference on Communications in China-Workshops (CIC/ICCC)*, pages 62–66. IEEE, 2015.
- [137] J. Ma, N. Lu, and H. Zhang. PSO-Based Proactive Routing in Delay Tolerant Network. In *International Conference on Cyberspace Technology (CCT)*, pages 1–4, Nov. 2014.
- [138] N. Maisonneuve, M. Stevens, M. Niessen, and L. Steels. Noisetube: Measuring and mapping noise pollution with mobile phones. pages 215–228, 01 2009.

- [139] T. Malche, P. Maheshwary, and R. Kumar. Environmental monitoring system for smart city based on secure internet of things (iot) architecture. *Wireless Personal Communications*, 107(4):2143–2172, 2019.
- [140] D. Marikyan, S. Papagiannidis, and E. Alamanos. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, 138:139–154, 2019.
- [141] J. Matute. “Publicly-Accessible Public Transportation Data”, 2018.
- [142] T. W. McLain and R. W. Beard. Coordination variables, coordination functions, and cooperative-timing missions. *Journal of Guidance, Control, and Dynamics*, 28, 2005.
- [143] Measure a 32 Advisor Company. Drones for Disaster Response and Relief Operations, Apr. 2015. <https://tinyurl.com/yy2pp6tc>.
- [144] M. S. Mekala and P. Viswanathan. A survey: Smart agriculture iot with cloud computing. In *2017 international conference on microelectronic devices, circuits and systems (ICMDCS)*, pages 1–7. IEEE, 2017.
- [145] A. Mitra, B. Bera, and A. K. Das. Design and testbed experiments of public blockchain-based security framework for iot-enabled drone-assisted wildlife monitoring. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2021.
- [146] F. Montori, L. Bedogni, and L. Bononi. A collaborative internet of things architecture for smart cities and environmental monitoring. *IEEE Internet of Things Journal*, 5(2):592–605, 2017.
- [147] 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>.
- [148] N. H. Motlagh, M. Bagaa, and T. Taleb. Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134, 2017.
- [149] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Mobile internet of things: Can uavs provide an energy-efficient mobile architecture? In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.
- [150] A. Mukhopadhyay et al. An online decision-theoretic pipeline for responder dispatch. In *ACM/IEEE Intel. Conference on Cyber-Physical Systems*, 2019.
- [151] S. Munirathinam. Industry 4.0: Industrial internet of things (iiot). In *Advances in computers*, volume 117, pages 129–164. Elsevier, 2020.
- [152] A. Nadi and A. Edrisi. Adaptive multi-agent relief assessment and emergency response. *Intel. journal of disaster risk reduction*, 24, 2017.

- [153] T. Nägeli et al. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 2017.
- [154] 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.
- [155] 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>,.
- [156] National Institute of Justice. A Guide for Investigating Fire and Arson, May 2009. <https://tinyurl.com/yy48k9wq>.
- [157] Netage B.V. Smart Data for Smarter Firefighters. <https://netage.nl/resc-info/>.
- [158] F. Nex and F. Remondino. Preface: Latest developments, methodologies, and applications based on uav platforms. *Drones*, 3(1), 2019.
- [159] T. Ni, W. Li, D. Zhao, and Z. Kong. Road profile estimation using a 3d sensor and intelligent vehicle. *Sensors*, 20(13):3676, 2020.
- [160] E. Niforatos, A. Vourvopoulos, M. Langheinrich, P. Campos, and A. Doria. Atmos: A hybrid crowdsourcing approach to weather estimation. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, page 135–138, New York, NY, USA, 2014. Association for Computing Machinery.
- [161] I. K. Nikolos et al. Uav path planning using evolutionary algorithms. *Innovations in intelligent machines-1*, 2007.
- [162] T. Oda, A. Barolli, E. Spaho, L. Barolli, and F. Xhafa. Analysis of Mesh Router Placement in Wireless Mesh Networks Using Friedman Test. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 289–296, Jun. 2014.
- [163] S. O’Dea. Data volume of iot connected devices worldwide 2019 and 2025, 2020.
- [164] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36, 2009.
- [165] J. Parkman. “6 Urban Green Space Projects That Are Revitalizing U.S. Cities”, 2016.
- [166] P. Pecho, P. Magdolenová, and M. Bugaj. Unmanned aerial vehicle technology in the process of early fire localization of buildings. *Transportation Research Procedia*, 40:461–468, 2019.
- [167] A. Petz, J. Enderle, and C. Julien. A Framework for Evaluating DTN Mobility Models. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 94:1–94:8, May 2009.

- [168] H. Pham, H. M. La, D. Feil-Seifer, and M. Deans. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In *Int. Conf. on Intel. Robots and Systems*, 2017.
- [169] M. Pierzchała, P. Giguère, and R. Astrup. Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam. *Computers and Electronics in Agriculture*, 145:217–225, 2018.
- [170] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research*, 31(12):1985–2002, 2004.
- [171] T. Qiu, K. Zheng, M. Han, C. P. Chen, and M. Xu. A data-emergency-aware scheduling scheme for internet of things in smart cities. *IEEE Transactions on Industrial Informatics*, 14(5):2042–2051, 2017.
- [172] L. Quintero-Cano, M. Wahba, and T. Sayed. Bus Networks as Graphs: New Connectivity Indicators with Operational Characteristics. *Canadian Journal of Civil Engineering*, Sep. 2014.
- [173] C. Raffelsberger and H. Hellwagner. A Hybrid MANET-DTN Routing Scheme for Emergency Response Scenarios. In *International Conference on Pervasive Computing and Communications (PerCom) Workshop*, Mar. 2013.
- [174] K. Rafiq, B. J. Pitcher, K. Cornelsen, K. W. Hansen, A. J. King, R. G. Appleby, B. Abrahms, and N. R. Jordan. Animal-borne technologies in wildlife research and conservation. *Conservation Technology*, pages 105–28, 2021.
- [175] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: An end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, page 105–116. Association for Computing Machinery, 2010.
- [176] R. Ravi and A. Sinha. Approximation Algorithms for Problems Combining Facility Location and Network Design. *Operations Research*, 54(1):73–81, Feb. 2006.
- [177] C. Reardon and J. Fink. Air-ground robot team surveillance of complex 3D environments. *Intl. Symp. on Safety, Security and Rescue Robotics*, pages 320–327, 2016.
- [178] H. U. Rehman, M. Asif, and M. Ahmad. Future applications and research challenges of iot. In *2017 International conference on information and communication technologies (ICICT)*, pages 68–74. IEEE, 2017.
- [179] D. Reifsteck, T. Engesser, R. Mattmüller, and B. Nebel. Epistemic multi-agent planning using monte-carlo tree search. In *KI 2019: Advances in Artificial Intelligence: 42nd German Conference on AI, Kassel, Germany, September 23–26, 2019, Proceedings 42*, pages 277–289. Springer, 2019.
- [180] Report Linker. “Internet of Things (IoT) Networks: Technologies and Global Markets to 2022”, 2017.

- [181] D. Roberts. “A Fascinating New Scheme to Create Walkable Public Spaces in Barcelona”, 2017.
- [182] V. M. Rohokale, N. R. Prasad, and R. Prasad. A cooperative internet of things (iot) for rural healthcare monitoring and control. In *2011 2nd international conference on wireless communication, vehicular technology, information theory and aerospace & electronic systems technology (Wireless VITAE)*, pages 1–6. IEEE, 2011.
- [183] C. Roman and H. Singh. Consistency based error evaluation for deep sea bathymetric mapping with robotic vehicles. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 3568–3574. Ieee, 2006.
- [184] S. A. Sadat, J. Wawerla, and R. T. Vaughan. Recursive non-uniform coverage of unknown terrains for uavs. In *Conf. on Intel. Robots and Systems*, 2014.
- [185] T. Sakaki, M. Okazaki, and Y. Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE transactions on knowledge and Data Engineering*, 25(4):919–931, 2012.
- [186] A. Salam. Internet of things for sustainable community development: introduction and overview. In *Internet of Things for sustainable community development*, pages 1–31. Springer, 2020.
- [187] A. Salam and A. Salam. Internet of things for sustainable forestry. *Internet of Things for sustainable community development: Wireless communications, sensing, and systems*, pages 147–181, 2020.
- [188] Sarah Calams. 6 takeaways on how fire departments are using drones and the barriers preventing purchase, 2018. <https://tinyurl.com/y2qhs43x>,.
- [189] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar. System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization. *Autonomous Robots*, 41:919–944, 2017.
- [190] T. Savolainen, J. Soinen, and B. Silverajan. Ipv6 addressing strategies for iot. *IEEE Sensors Journal*, 13(10):3511–3519, 2013.
- [191] M. Schaefer, R. Teeuw, S. Day, D. Zekkos, P. Weber, T. Meredith, and C. J. Van Westen. Low-cost uav surveys of hurricane damage in dominica: automated processing with co-registration of pre-hurricane imagery for change analysis. *Natural hazards*, 101(3):755–784, 2020.
- [192] K. Schmid, P. Lutz, T. Tomić, E. Mair, and H. Hirschmüller. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, 31(4):537–570, 2014.



- [193] E. Schubert and P. J. Rousseeuw. Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms. In G. e. a. Amato, editor, *Similarity Search and Applications*. Springer Intl. Publishing, 2019.
- [194] 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*, 10848 LNCS, 2018.
- [195] M. Sede, X. Li, D. Li, M. Wu, M. Li, and W. Shu. Routing in Large-Scale Buses Ad Hoc Networks. In *WCNC*, pages 2711–2716, Mar. 2008.
- [196] J. Shah and B. Mishra. Iot enabled environmental monitoring system for smart cities. In *2016 international conference on internet of things and applications (IOTA)*, pages 383–388. IEEE, 2016.
- [197] A. Sharif, J. Li, M. Khalil, R. Kumar, M. I. Sharif, and A. Sharif. Internet of things—smart traffic management system for smart cities using big data analytics. In *2017 14th international computer conference on wavelet active media technology and information processing (ICCWAMTIP)*, pages 281–284. IEEE, 2017.
- [198] R. Sharp and H. Peng. Vehicle dynamics applications of optimal control theory. *Vehicle System Dynamics*, 49(7):1073–1111, 2011.
- [199] 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*, pages 0–0, 2019.
- [200] S. L. Smith and D. Rus. Multi-robot monitoring in dynamic environments with guaranteed currency of observations. *IEEE Conf. on Decision and Control*, 2010.
- [201] C. Song, J. Gu, and M. Liu. Deployment Mechanism Design for Cost-Effective Data Uploading in Delay-Tolerant Crowdsensing. In *IEEE International Symposium on Parallel and Distributed Processing with Applications and IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Dec. 2017.
- [202] Q. Song and e. a. Zeng, Y. A survey of prototype and experiment for uav comms. *Science China Information Sciences*, 2021.
- [203] J. A. Stankovic and e. a. M. Spuri. *Deadline scheduling for real-time systems: EDF and related algorithms*. Springer Science & Business Media, 1998.
- [204] B. L. R. Stojkoska and K. V. Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of cleaner production*, 140:1454–1464, 2017.
- [205] M. Štolba and A. Komenda. The madla planner: Multi-agent planning by combination of distributed and local heuristic search. *Artificial Intelligence*, 252:175–210, 2017.
- [206] K. Su, J. Li, and H. Fu. Smart city and the applications. In *2011 international conference on electronics, communications and control (ICECC)*, pages 1028–1031. IEEE, 2011.

- [207] S. Suzuki. Recent researches on innovative drone technologies in robotics field. *Advanced Robotics*, 32(19):1008–1022, 2018.
- [208] S. Tanwar, S. Tyagi, and S. Kumar. The role of internet of things and smart grid for the development of a smart city. In *Intelligent Communication and Computational Technologies: Proceedings of Internet of Things for Technological Development, IoT4TD 2017*, pages 23–33. Springer, 2018.
- [209] Team Recurrency. The IoT Data Explosion: How Big Is the IoT Data Market?, January 2019. <https://priceconomics.com/the-iot-data-explosion-how-big-is-the-iot-data/>.
- [210] S. R. Thangiah, J.-Y. Potvin, and T. Sun. Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, 23(11):1043–1057, 1996.
- [211] A. Torreno et al. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)*, 50, 2017.
- [212] A. Torreno, E. Onaindia, A. Komenda, and M. Štolba. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)*, 50(6):1–32, 2017.
- [213] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *Inform's Journal on computing*, 15(4):333–346, 2003.
- [214] E. E. Tsiropoulou, S. T. Paruchuri, and J. S. Baras. Interest, Energy and Physical-Aware Coalition Formation and Resource Allocation in Smart IoT Applications. In *CISS*, pages 1–6, Mar. 2017.
- [215] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 108–112, 2013.
- [216] E. Tuyishimire, A. Bagula, S. Rekhis, and N. Boudriga. Cooperative data muling from ground sensors to base stations using uavs. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 35–41. IEEE, 2017.
- [217] S. L. Ullo and G. R. Sinha. Advances in smart environment monitoring systems using iot and sensors. *Sensors*, 20(11):3113, 2020.
- [218] United States Fire Administration National Fire Academy. Incident Command for Highrise Operations ICHO-Student Manual. 2006.
- [219] K. P. Valavanis and G. J. Vachtsevanos. *Handbook of unmanned aerial vehicles*. Springer, 2015.
- [220] 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.*, 12, 2015.

- [221] Z. Wang et al. Multi-agent based path planning for first responders among moving obstacles. *Computers, Environment and Urban Systems*, 56, 2016.
- [222] N. White and M. Delichatsios. *Fire hazards of exterior wall assemblies containing combustible components*. Springer, 2015.
- [223] Wildland Fire Lessons Learned Center. North Schell Escaped RX (2012). <https://tinyurl.com/bdddnksc>.
- [224] G. Wilkinson. Digital terrestrial tracking: The future of surveillance. *DEFCON*, 22, 2014.
- [225] H. Wirtz, J. R uth, M. Serror, J.  . Bitsch Link, and K. Wehrle. Opportunistic interaction in the challenged internet of things. In *Proceedings of the 9th ACM MobiCom workshop on Challenged networks*, pages 7–12, 2014.
- [226] F. Xhafa, C. S nchez, A. Barolli, and M. Takizawa. Solving Mesh Router Nodes Placement Problem in Wireless Mesh Networks by Tabu Search Algorithm. *Journal of Computer and System Sciences*, 81(8), Dec. 2015.
- [227] F. Xhafa, C. S nchez, and L. Barolli. Locals Search Algorithms For Efficient Router Nodes Placement in Wireless Mesh Networks. In *International Conference on Network-Based Information Systems (NBiS)*, Aug. 2009.
- [228] X. Xiao, B. Liu, G. Warnell, and P. Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022.
- [229] J. Xin, H. Zhao, D. Liu, and M. Li. Application of deep reinforcement learning in mobile robot path planning. In *2017 Chinese Automation Congress (CAC)*, pages 7112–7116. IEEE, 2017.
- [230] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier. CrowdTasker: Maximizing Coverage Quality in Piggyback Crowdsensing Under Budget Constraint. In *International Conference on Pervasive Computing and Communications (PerCom)*, Jul. 2015.
- [231] K. Xiong. Solving the Performance Puzzle of DSRC Multi-Channel Operations. In *IEEE International Conference on Communications (ICC)*, Jun. 2014.
- [232] Z. Xu, L. Mei, K.-K. R. Choo, Z. Lv, C. Hu, X. Luo, and Y. Liu. Mobile crowd sensing of human-like intelligence using social sensors: A survey. *Neurocomputing*, 279:3–10, 2018. Advances in Human-like Intelligence towards Next-Generation Web.
- [233] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on Autonomous agents*, pages 616–623, 2001.

- [234] G. Yang and X. e. a. Lin. A telecom perspective on the internet of drones: From lte-advanced to 5g. *arXiv preprint*, 2018.
- [235] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, oct 2016.
- [236] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.
- [237] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris. A review of machine learning and iot in smart transportation. *Future Internet*, 11(4):94, 2019.
- [238] M. Zarafshan-Araki and K.-W. Chin. Trainnet: A transport system for delivering non real-time data. *Computer Communications*, 33(15):1850–1863, 2010.
- [239] Y. Zguira, H. Rivano, and A. Meddeb. Iob-dtn: A lightweight dtn protocol for mobile iot applications to smart bike sharing systems. In *2018 Wireless Days (WD)*, pages 131–136. IEEE, 2018.
- [240] F. Zhang, B. Jin, Z. Wang, H. Liu, J. Hu, and L. Zhang. On Geocasting over Urban Bus-Based Networks by Mining Trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1734–1747, Jun. 2016.
- [241] F. Zhang, H. Liu, Y. Leung, X. Chu, and B. Jin. CBS: Community-Based Bus System as Routing Backbone for Vehicular Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 16(8):2132–2146, Aug. 2017.
- [242] S. Zhang, D. Gao, H. Lin, and Q. Sun. Wildfire detection using sound spectrum analysis based on the internet of things. *Sensors*, 19(23):5093, 2019.
- [243] M. Zhao, Y. Yang, and C. Wang. Mobile Data Gathering with Load Balanced Clustering and Dual Data Uploading in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 14(4):770–785, Apr. 2015.
- [244] S. Zhao, Q. Wang, X. Fang, W. Liang, Y. Cao, C. Zhao, L. Li, C. Liu, and K. Wang. Application and development of autonomous robots in concrete construction: Challenges and opportunities. *Drones*, 6(12):424, 2022.
- [245] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.
- [246] K. Zhu and T. Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.
- [247] Q. Zhu, W. Chen, H. Hu, X. Wu, C. Xiao, and X. Song. Multi-sensor based attitude prediction for agricultural vehicles. *Computers and electronics in agriculture*, 156:24–32, 2019.

- [248] Q. Zhu, M. Y. S. Uddin, Z. Qin, and N. Venkatasubramanian. Upload Planning for Mobile Data Collection in Smart Community Internet-of-Things Deployments. In *IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, 2016.