UNIVERSITY OF LEEDS

Distributed Adaptive Multi-Drone Coordination At Scale

Chuhao Qin

University of Leeds

School of Computer Science

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

To Zhēn and Yù,
who have taught me that love's truest name is Precious.
One gave me Life, the other made it Shine.
This thesis is my Treasure–Yours.

Intellectual Property Statement

Acknowledgements

Pursuing a PhD demands patience. Indeed, bridging theory and practice is a long and intricate journey, one where pleasure and pain intertwine. Along this path, I overcame countless obstacles, striving to hear the music of science. None of this would have been possible without the support of several people. To them, I am sincerely grateful for this.

First and foremost, I would like to thank my supervisor, Evangelos Pournaras, for bringing me to the University of Leeds and giving me careful guidance. His source of knowledge, inspiration, and innovative thinking have profoundly shaped my academic growth. During my PhD, Vangelis and I went through countless discussions, each helping me to spark new insights. His rigorous scholarship, relentless pursuit of answers, and meticulous revisions pushed me to refine my research and writing, making this thesis possible. I still remember his words, *"Think as a scientist, not as an engineer,"* which will forever guide my future academic career. It is my luck and honor to be his first PhD student at Leeds.

It has been my pleasure to have many great colleagues in DSS group at University of Leeds. Though we came from distant corners of the world, our differences only strengthened our bonds. Through weekly lunch meeting, hiking, and countless conversations, we exchanged cultural traditions, personal hobby, and intellectual perspectives with much enthusiasm. These moments not only broadened my horizons but brought warmth and vitality to my solitary pursuit of research. Specially, I want to thank to Srijoni, an extraordinary Indian scholar whose brilliance is matched only by her devotion as a mother. Her ability to balance heavy research workload with family life inspired me, and our daily academic discussion have profoundly shaped my work. I also extend my sincere appreciation to research fellows from Leeds, Sheffield, EPFL, and LUT universities whose collective wisdom has illuminated my PhD journey.

To Mingyu Ding, you are my love, my partner, and my soulmate. Our paths crossed in London, and since that day, two years of shared journey have unfolded. Your light always brighten my world, eliminating the fatigue of long hours in research. Your keen mind and quick wit have guided me through moments of doubt, offering clarity when I was lost and anxious.

You are the treasure I found in this foreign land. *You complete me.*

Last but not least, I wish to express my gratitude to my parents back in China. You have been my most unwavering pillars of support throughout this hard journey toward a PhD. Though half a world apart, no distance could ever sever the invisible bond that connects us. I am forever grateful for your sincere comfort and encouragement during moments of exhaustion, reminding me that home will always be my ultimate sanctuary, and familial love the most precious of all human connections. No words could ever fully convey my appreciation, so let me borrow this line from my favorite film as tribute: *"You are the reason I am. You are all my reasons."*

Abstract

Designing and understanding the multi-drone operations is a grand challenge. This is particularly prominent in intelligent transportation systems where swarms of cooperative drones are used for traffic monitoring and last-mile delivery. Although significant technological breakthroughs have been achieved in the control and communication of individual drones, coordinating multiple drones for distributed task allocation remains an open research problem. This involves determining which drones should visit which points of interest on the map, and when, to execute tasks, such as collecting sensing data and inspecting infrastructure, thereby maximizing mission performance (e.g., high completion rate, accurate sensing). Figuring out the best way to allocate these tasks is complex and falls into a class of problems known as NP-hard combinatorial optimization.

Previous work addresses this problem by employing distributed task allocation algorithms, from market-based methods to swarm intelligence. However, such approach comes with three key limitations: (1) *Poor scalability in large-scale operations*: Existing approaches struggle with long-term, large-scale planning due to high computational and communication costs. (2) *Poor adaptability to diverse real-world conditions*: Current models often overlook or over-simplify real-world factors, such as drone weight, payload and recharging locations, making it difficult to estimate energy consumption and adapt to environmental dynamics. (3) *Costly, risky and oversimplified prototyping*: Even scalable and adaptive models need real-world testing with low-cost, safe but not oversimplified prototyping to disentangle the complexity of multi-drone coordination before real-world deployment.

To address these challenges, this thesis contributes to propose a distributed multi-agent coordination model where each agent/drone independently determines navigation, tasking and recharging plans to choose from such that system-wide requirements are met. The model optimizes the discrete plan selection by integrating state-of-the-art reinforcement learning, collective learning and exact algorithms, which enhances both scalability and adaptability in evolving environments under critical hard constraints. As a proof of concept, this work focuses on two scenarios: traffic monitoring in urban sensing and last-mile delivery in logistics. Experimental results demonstrate that the proposed method achieves scalable, energy-efficient and accurate task execution in large-scale spatio-temporal scenarios (e.g., coordinating 1,000 drone dispatches for full-day, city-sized missions), while operating under limited drone and energy resources. These findings offer valuable insights for policymakers, system operators, and technology designers aiming to enhance intelligent transportation systems. Potential benefits include reducing carbon emissions from heavy vehicles, alleviating traffic congestion, improving the quality of urgent medical deliveries (e.g., pandemics), and informing the optimal placement of recharging stations or depots.

Furthermore, the development of an indoor testbed ensures low-cost deployment, operational safety, and the applicability of the proposed system. These advantages are validated through hardware experiments, which confirm accurate energy consumption estimation and minimal collision risk. The proposed testbed bridges the gap between complex algorithmic simulations and practical but oversimplified multi-drone implementations before outdoor testing. It provides a replicable and scalable prototyping, paving the way for broader adoption of advanced multi-drone systems.

Contents

Chapter 1

Introduction

The rapid advancement and adoption of autonomous aerial systems, particularly Unmanned Aerial Vehicles (UAVs) commonly known as *drones*, have opened new opportunities for innovation in intelligent transportation systems [1, 2]. The integration of drone technologies into various sectors has accelerated significantly in recent years, driven by its potential to enhance operational efficiency and reduce environmental impact. According to recent findings[1] from the UK's National Aeronautical Centre, 42% of carriers in Europe are actively exploring the incorporation of drones into their operations. This growing interest is expected to generate substantial economic impact, with projections estimating the creation of approximately 100,000 jobs directly linked to the drone industry in the near future.

However, compared to relying on a single high-performance drone, which is often expensive and constrained by limited flight range due to the battery constraints, a swarm of low-cost drones offers a more flexible and scalable solution [3]. This collective capability assists drones to divide and perform tasks in parallel, enhancing system-wide efficiency. The tasks encompass a variety of missions in transportation, including sensing data collection (e.g., video, temperature and humidity) for traffic monitoring or crop inspection, and target tracking for disaster response or last-mile delivery. Taking an example of traffic monitoring, a drone swarm equipped with cameras can be deployed to observe different sections of road network in a city simultaneously [4]. While one drone captures traffic flow at a congested intersection, others monitor adjacent roads or respond to sudden incidents such as accidents or roadblocks. This distributed coverage

[1]Nine Jobs in the Drone Industry (With Salaries and Primary Duties), https://www.indeed.com/career-advice/finding-a-job/jobs-in-the-drone-industry

allows for faster detection of traffic anomalies and supports real-time data collection for urban mobility management [4].

Therefore, the distributed coordination is required for drone swarms, which is a process of organizing multiple agents/drones to collaboratively work together efficiently toward a common goal of optimal task allocation while considering their own cost (e.g., energy consumption of a trip). The problem is formulated as a NP-hard multi-objective combinatorial optimization problem, aiming to determine how to allocate tasks (e.g., sensing data collection, delivery) to individual drones, and how to plan their flight paths [5]. Effective coordination minimizes redundant travel and low-priority coverage, leading to better drone resource utilization and lower energy consumption. This enables more tasks to be completed with fewer drones, reducing operational costs and enhancing mission performance, which makes this problem both critical and timely. Nevertheless, solving the problem has various requirements determined by specific task scenarios. For instance, every customer in a delivery mission must be delivered exactly once, which restricts drone from repeatedly visiting or missing the same point of interest. When the systems are evolving (e.g., the unpredictable traffic flow or delivery requests), drones must continuously interact with environment and improve their behavior of navigation and tasking to maximize long-term mission performance, such as accurate sensing and low delivery delay. To address these challenges, a variety of task allocation algorithms have been developed. They range from exact algorithms [6], which guarantee optimal solutions under hard constraints, to heuristic methods such as swarm intelligence [3, 7] and multi-agent reinforcement learning (MARL) [8], which encourage drones to individually explore optimal behaviors of navigation and tasking in dynamic environments.

Despite significant advances in distributed multi-drone coordination, most existing methods face the following three obstacles: (1) *Poor scalability in large-scale operations*: Assigning tasks to drones when they are working across wide areas and over a long time span (e.g., an full-day span) leads to more possible decisions of navigation and tasking for each drone. Searching the optimal solution requires high computation and communication cost using traditional methods [9, 10]. (2) *Poor adaptability to diverse real-world conditions*: In real-world scenarios (e.g., traffic monitoring, delivery), drones must operate under constantly changing conditions, including dynamic task requirements, different payloads, different recharging locations, and heterogeneous drone capabilities (e.g., size, weight). These factors significantly affect drone performance, especially energy consumption, while most existing models assume static or idealized conditions and fail to adapt to environmental dynamics. (3) *Costly, risky and over-simplified prototyping*: A big gap remains between simulation results and practical deployment. While simulations are useful for evaluating complex task allocation of swarms in scalable and adaptive coordination, existing testbed prototypes typically support only simple flight patterns due to hardware technical challenges such as high cost and collision avoidance.

Therefore, a scalable and adaptive approach is needed, one that starts from the systematic design of distributed multi-agent coordination models, to real-world prototyping, testing and validation through a versatile testbed experimentation. By bridging theoretical modeling with physical implementation, this approach advances our understanding of how drone swarms can make adaptive, energy-aware decisions in evolving environments, moving beyond idealized simulations toward solutions grounded in physical and operational constraints.

1.1 Research Scope

This thesis focuses on the high-level coordination of multi-drone systems, particularly addressing the challenges of task allocation in real-world, civilian, transportation applications, including traffic management, disaster response, smart parking, and last-mile delivery [4, 11–13]. It emphasizes strategic decision-making that assists drones to navigate, execute tasks and recharge effectively in diverse, mission-critical environments [5]. The goal is to empower scalable and distributed coordination among a swarm of drones operating over large spatio-temporal missions (e.g., data collection over a 2D city map within a whole day). While drone-related technologies are widely used in edge/cloud-assisted networks, UAV-to-Terrestrial/Satellite communication, and optimized landing and takeoff [14], this work deliberately does not concentrate on low-level communication or control theory.

The core problem targeted by this thesis lies in the distributed task allocation. The problem seeks to find the most efficient set of routes and tasks for autonomous agents (or drones) in a distributed manner, allowing each agent to make decisions independently. Thus, this problem is mathematically formulated as a *NP-hard combinatorial optimization problem* [15], closely related to classical problems such as the traveling salesman problem and 0-1 knapsack problem. To tackle the problem, this thesis targets on *distributed optimization* algorithms, such as consensus-based bundle algorithm, as well as the distributed variants of traditional heuristic approaches (e.g., genetic algorithm, particle swarm optimization) [9, 10, 16]. Notably, the multi-agent collective learning [17, 18] has emerged as a promising approach for addressing this problem. It enables self-organized, decentralized coordination among agents, offering high scalability (supporting a large number of agents) and efficiency (with minimal communication and computational overhead), while preserving individual agent autonomy to adapt their decisions dynamically. Apart from these *short-term* algorithms, this thesis explores the *long-term* benefits of learning algorithms (e.g., MARL), especially in dynamic or uncertain environments [16, 19]. Here, the centralized training decentralized execution mode allows agents to learn optimal policies efficiently during training, while operating independently with only local observations during execution. To solve problems with both *soft* and *hard* constraints, traditional exact algorithms (e.g., greedy algorithm, branch-and-bound) [20] are considered in this thesis.

The scope further extends to real-world drone-based scenarios, where the abstract models and techniques are instantiated into specific, high-impact applications. Two key application domains within intelligent transportation systems are explored: (1) urban sensing and (2) drone logistics. In urban sensing, drones are equipped with advanced sensing modules such as LiDAR, cameras, or environmental sensors (e.g., for temperature and humidity) to collect diverse types of data across a city [21]. These drones must be intelligently self-assigned to points of interest to ensure energy-efficient and accurate data collection, tailored to the specific sensing objectives. A typical example is traffic monitoring, where camera-equipped drones observe and analyze vehicle flows, detect accidents or congestion, and relay timely insights to traffic operators for timely interventions [4, 22]. In drone logistics, drones deliver parcels to traffic-congested or hard-to-reach areas, thereby reducing reliance on ground vehicles, alleviating congestion, shortening delivery times, and reduce carbon emissions [13, 23]. Drones in this context aim to minimize delivery delay and optimize energy usage, enhancing both service quality and logistics efficiency. These two representative scenarios are selected because they contain a wide range of real-world complexities, including static and evolving systems, scarce and sufficient drone resources, as well as both soft (e.g., user preferences, task completion) and hard (e.g., battery limits) constraints. As a result, they provide a robust testing ground to validate the generality, scalability, and adaptability of the proposed research approach in this thesis.

Finally, to ensure realistic and practical validation of the multi-agent coordination model, this thesis focuses on indoor experimentation using low-cost drone platforms. Outdoor drone testing, especially with multiple drones, often faces challenges such as high cost, regulatory restrictions, unpredictable weather, and safety risks, which can hinder iterative development and experimentation [24]. In contrast, indoor environments offer controlled, repeatable, and scalable testing conditions, enabling efficient evaluation of the proposed model under various constraints [25–27]. Moreover, operating in confined spaces brings collision avoidance to the forefront as a critical consideration, making it an integral part of the system design.

1.2 Research Objectives

The research objectives of this thesis mainly address the following research question:

Overarching Research Question: *How to design and prototype a distributed multi-drone coordination system that supports scalable and adaptive task allocation across diverse applications?*

To answer the above research question, this thesis mainly designs a distributed model that enables drones to autonomously generate energy-aware plans for navigation, task execution, and recharging. Each plan consists of a sequence of tasks scheduled within the maximum flight time of drones, which reduces the decision space for each agent and improves scalability. By incorporating real-time task demands and estimated energy consumption, the model allows drones to self-adapt their decisions to dynamic and complex environments, enhancing both adaptability and efficiency. The superior performance in both scalability and adaptability of the model is validated by extensive experiments, e.g., varying the number of drones and task requirements. Furthermore, three sub-objectives are summarized in the rest of this section, each corresponding to a single one research question.

1.2.1 Coordination at scale

The first research question is formulated as:

Research Question 1: *How can a distributed multi-drone coordination system* *be designed to scale effectively in large spatio-temporal environments while ensuring long-term benefits and satisfying hard constraints?*

Traditional approaches to multi-objective combinatorial optimization, such as multi-agent collective learning, are relevant to large-scale systems due to their low computational and communication overhead. The system is efficient even scaling to high number of agents that perform tasks across wide areas. Such a scale in task allocation is essential for effective city-wide traffic and logistic management in advance. However, these methods focus on short-term optimization and are often hand-crafted for specific, static scenarios. When extended to long time spans (e.g., full-day sensing or delivery task), they struggle to anticipate future consequences and adapt to changing conditions. In contrast, MARL offers a framework for sequential decision-making, enabling agents to learn policies that optimize long-term outcomes under dynamic conditions. Nevertheless, as the number of agents increases, MARL faces critical challenges: exponential growth of the joint state-action space, increased communication cost, and privacy concerns associated with centralized training. Furthermore, existing distributed optimization methods often lack robust mechanisms for enforcing both soft and hard constraints, which are essential in real-world multi-drone deployments.

To bridge the conceptual gap, this thesis designs a generic distributed multi-agent coordination model that integrates the strengths of short-term distributed optimization, long-term learning and hard constraint satisfaction. The objective here is to minimize both agent- and system-level costs while satisfying hard constraints on the system. The architecture of this model, including its core components, their interactions and inter-relations, are defined and studied in Chapter 3.

1.2.2 Drone-based adaptability

The second research question is formulated as:

Research Question 2: *How can a distributed multi-drone coordination system be designed to adapt to diverse and dynamic real-world conditions such as energy constraints?*

Existing distributed multi-drone coordination models [28–30] for task allocation typically focus on minimizing travel time or distance, often overlooking the critical aspect of battery constraints and energy consumption. In reality, energy use depends on a range of complex and interdependent factors, including drone capabilities (e.g., weight, propeller and motor efficiency), environmental conditions (e.g., wind speed, temperature), and task-specific requirements (e.g., payload weight during delivery). Neglecting these factors leads to inaccurate estimates of energy needs and suboptimal task performance. Moreover, energy-awareness is essential for safety and compliance. UAV safety guidelines recommend that missions should be completed when battery levels reach around $25\% - 30\%$, to avoid unexpected depletion and ensure safe landing. However, most existing models [28–30] fail to incorporate dynamic energy constraints or plan for periodic recharging, especially over full-day missions that span multiple time intervals. This results in drones lacking a planned destination for recharging or landing, which can limit their future task capacity and reduce the reliability of long-term operations.

Therefore, this thesis designs two system models based on real multi-drone applications: The first model, detailed in Chapter 4, focuses on urban sensing. It tackles the challenge of distributed coordination among heterogeneous drones for energy-efficient and accurate data collection, such as vehicle observation across urban environments. It accounts for numerous and heterogeneous drones performing different types of sensing operations over a broad geographic area. The second model, presented in Chapter 5, addresses the time-sensitive last-mile delivery problem with energy-efficient, multi-parcel route planning. It considers delivery deadlines and payload constraints, where energy estimation becomes especially complex due to changing parcel weights and varying delivery paths. Both models simulate full-day operations, where time is divided into multiple periods constrained by battery life. At the end of each period, drones must strategically select a recharging location to maintain long-term task performance across the mission timeline. This approach allows for energy-aware, adaptive decision-making in diverse real-world scenarios.

1.2.3 Testbed experimentation

The third research question is formulated as:

Research Question 3: *How to prototype the distributed multi-drone coordination system to be low costly, low risky, and easy to manage, reducing fragmentation of simulation and real-world applications?*

Most existing research on distributed multi-drone coordination relies heavily on simulation, which, while useful for testing algorithms safely and at low cost, often fails to capture critical real-world complexities such as noisy sensor data, non-ideal energy consumption, and physical constraints (e.g., collisions and wind resistance). In addition, current physical testbeds typically support only simple or small-scale flight scenarios, lacking the scale and realism needed to evaluate complex, long-term, and energy-aware coordination algorithms. This creates a gap between algorithmic performance in theory and actual operational feasibility in practice.

To address this gap, Chapter 6 focuses on prototyping indoor testbed to emulate the outdoor task environments with high realism, including dynamic targets and battery constraints. The cheap, tiny and easily programmable hardware drones are adopted to simplify the testbed design, testing and evaluation. Particular attention is given to safety requirements, especially collision avoidance (i.e., low risk of collisions), which becomes a critical issue when multiple drones operate in confined indoor spaces.

1.3 Research Approach

Achieving the research objectives of this thesis requires a comprehensive understanding of inter-disciplinary domains, from computer science to robotics, including the design of generic model for application-driven multi-drone task allocation (Section 1.2.1), coordination models for urban sensing and drone logistics (Section 1.2.2), and experimental validation through testbed prototype (Section 1.2.3). This section outlines the research approach adopted to address the overarching research question.

1.3.1 Research Philosophy

This thesis primarily adopts the *positivism*, a research philosophy, which asserts that scientific knowledge should be derived from objective, observable, and measurable reality [31]. In line with this view, experimental data is collected systematically to establish empirical evidence, and the research process emphasizes rigor reproducibility and transparency. Positivism is particularly suitable for studies involving large, distributed, and complex systems, such as the multi-drone coordination investigated in this thesis, where rigorous experimentation and quantitative analysis are conducted.

In addition, this work incorporates elements of *interpretivism* as a secondary research philosophy [31]. Interpretivism recognizes that knowledge is often shaped by context and subjectivity, particularly in scenarios involving human or environmental interaction. It allows for the construction of multiple contextual realities (e.g., varied experimental setups) to better understand phenomena. This perspective is reflected in parts of the research where data is shaped by scenario-specific interpretations, such as the traffic streams datasets in Chapter 4 and the data of delivery requests used in Chapters 5.

1.3.2 Research Strategy

The research strategy adopted in this thesis is *design science*, guided by the seven guidelines established by Hevner [32]. Design science aims to create purposeful and innovative artifacts[1] that address real-world problems. This thesis introduces four key artifacts/models: (1) a multi-agent coordination model (Section 1.4.1), (2) an energy-aware multi-drone sensing model (Section 1.4.2), (3) an energy- and delay-aware multi-drone delivery model (Section 1.4.3), and (4) a tasking experimentation testbed model (Section 1.4.4). Each model is rigorously evaluated using established methods to demonstrate its effectiveness and relevance, offering contributions to the field of distributed intelligent systems and their applications to swarms of drones.

Unlike behavioral science, which seeks to understand "what is true", design science focuses on developing "what is effective". It generates knowledge through the construction and contextual evaluation of artifacts. While primarily artifact-driven, design science also embraces behavioral aspects to assess an artifact's feasibility and utility in real-world settings. Finally, Chapter 7 reflects on the implications of this research for future collaborative decision-making systems and sustainable smart societies.

1.4 Contributions

The main contribution of this thesis is the development of *a novel, generic and highly efficient distributed multi-agent coordination model in large spatio-temporal environments* (Chapter 3, contribution 1 in Section 1.4.1). To demonstrate the effectiveness of this model, two new and energy-aware multi-drone coordination models are presented: one advancing traditional urban sensing (Chapter 4, contribution 2 in Section 1.4.2), and the other addressing the challenges of last-mile delivery (Chapter 5, contribution 3

[1]Constructs, models, methods and instantiations are general examples of artifacts in design science that refer to concepts, symbols, abstractions, representations, algorithms, prototypes, frameworks and etc [32].

in in Section 1.4.3). Additionally, this thesis contributes to a first working prototype of the testbed model to study distributed multi-drone coordination algorithms (Chapter 6, contribution 4 in in Section 1.4.4). All the material included in this thesis represents first-author contributions, with each chapter based on one or two scholarly papers. Three of these papers have been published in peer-reviewed international journals or conferences[1] [22, 26, 27], while the remaining works are under review [16, 23, 33].

1.4.1 Distributed multi-agent coordination model

To address the Research Question 1 in Section 1.2.1, this thesis presents a novel distributed multi-agent coordination model. It leverages a hierarchical framework to integrate a MARL algorithm for long-term decision-making in evolving multi-agent systems, a multi-agent collective learning algorithm for large-scale and privacy-preserving coordination of agents within static environments, as well as exact algorithms for adapting to hard environmental constraints. The proposed solution combines the strengths of these methods, ensuring scalability, decentralization, long-term effectiveness, and hard constraints satisfaction.

Chapter 3 also contributes: (1) Two novel high-level MARL-based strategies: (i) grouping task plan categories to reduce action space and (ii) constraining the agent behavior to enhance Pareto optimality. (2) Evaluation of how key factors influence performance, including the number of agents, the number of plans generated by each agent, and the complexity of target tasks, confirming the efficiency and scalability of the proposed approaches. (3) Insights about the optimality sacrifice as moving from soft to hard constraints and how this optimality loss is measured in terms of the required behavioral shift to preserve performance, i.e. restoring altruism deficit. (4) An open-source release of the code and algorithms[2] as well as a software artifact implementation of the hard constraint model[3] in order to ease the future research in this field.

[1]Chapter 4 in Transportation Research Part C: Emerging Technologies (IF=7.6, December 2023), and Chapter 6 in Advances in Computational Intelligence Systems (May 2024) and IEEE Conference on Local Computer Networks (October 2024).

[2]https://github.com/TDI-Lab/HRCL

[3]https://github.com/epournaras/EPOS

1.4.2 Coordination model for navigation and sensing

To address the Research Question 2 in Section 1.2.2, this thesis proposes a new energy-aware coordination of multi-drone navigation and sensing model, which is built based on the proposed distributed multi-agent coordination model. This is the the first study of task allocation for large-scale, dynamic urban sensing by a swarm of drones, focusing on optimizing the flight paths, data collection strategies, and recharging. The model aims to find optimal full-day navigation and sensing operations of a swarm of drones such that drones can efficiently and accurately collect required sensing data while minimizing their energy consumption.

Chapter 4 also contributes: (1) A plan generation strategy with three policies based on a power consumption model [34] to make the coordination model energy-aware and achieve a highly efficient navigation and sensing of drones. (2) A comprehensive empirical understanding of how a large spectrum of factors such as the number of dispatched drones, the drone density, the spatial granularity of sensing, the number of base stations, the number of time periods and the required amount of collected data influence sensing performance. (3) A testbed for extensive experimentation with realistic traffic patterns and real-world transport networks [4] under different vehicle density to validate efficient and accurate vehicle observation of the proposed approaches in both static and evolving systems. An open dataset [35] is generated containing all plans of the studied scenario.

1.4.3 Coordination model for last-mile delivery

Similarly to Section 1.4.2, this thesis proposes a new energy-and-delay-aware coordination of multi-drone last-mile delivery model. This is the first study of the task allocation for large spatio-temporal last-mile delivery by drone swarms accounting for both time-sensitive customer requests and energy consumption of multi-parcel route planning. The model aims to find optimal routes of a swarm of drones over all time windows such that the customers with different expected delivery time receive parcels with minimum time delay while reducing the energy consumption of drones.

Chapter 5 also contributes: (1) A decomposition approach to divide the whole delivery problem into three sub-problems, including a *delivery requests segmentation* problem, a *flight range selection* problem, and a *optimized plan selection* problem. (2)

A novel synthesis of algorithms[1] to combine K-means clustering, reinforcement learning and exact algorithms to strategically determine the flight range of drones for long-term delivery efficiency, while offloading the optimized plan selection to the exact algorithms and mixed integer programming solver [6]. (3) The validation of superior performance of the proposed approach over baseline methods through extensive experimentation using real-world delivery dataset, which provides new insights into a sustainable (low energy consumption and carbon emission), timely (low delivery delays), and adaptive (high operation speed) multi-drone delivery.

1.4.4 Indoor testbed prototype

To address the Research Question 3 in Section 1.2.3, this thesis propose a generic testbed model, a first working prototype of the testbed model with a proof-of-concept on accurate estimates of energy consumption and low risk of collisions in coordinated navigation and tasking. The testbed model aims to study task allocation and collision avoidance by a swarm of drones.

Furthermore, Chapter 6 contributes: (1) A new application domain of multi-agent collective learning on drone distributed sensing, as well as artificial potential field [36] for collision avoidance, using real traffic monitoring data [4]. (2) A rigorous evaluation of the testbed prototype in an indoor lab environment using real data to demonstrate its realism and highlight new insights into the low cost, safety and applicability of multi-drone task allocation. (3) An open-source software platform[2] and documentation[3] as a benchmark, providing detailed, reproducible coding examples and instructions for M-SET to foster future development and collaboration within the broader community.

1.5 Thesis Outline

This thesis is outlined in the six chapters illustrated in Fig. 1.1:

Chapter 2 positions the research of this thesis within related work on multi-drone task allocation.

Chapter 3 introduces the core system model, named *Planning-based Multi-Agent Coordination* (PMAC). It involves two approaches, the *Optimized Plan Selection* (*OPS*)

[1]The open source is available at: https://github.com/TDI-Lab/MAR-OPS.

[2]https://github.com/TDI-Lab/M-SET

[3]https://github.com/TDI-Lab/M-SET-Documentation

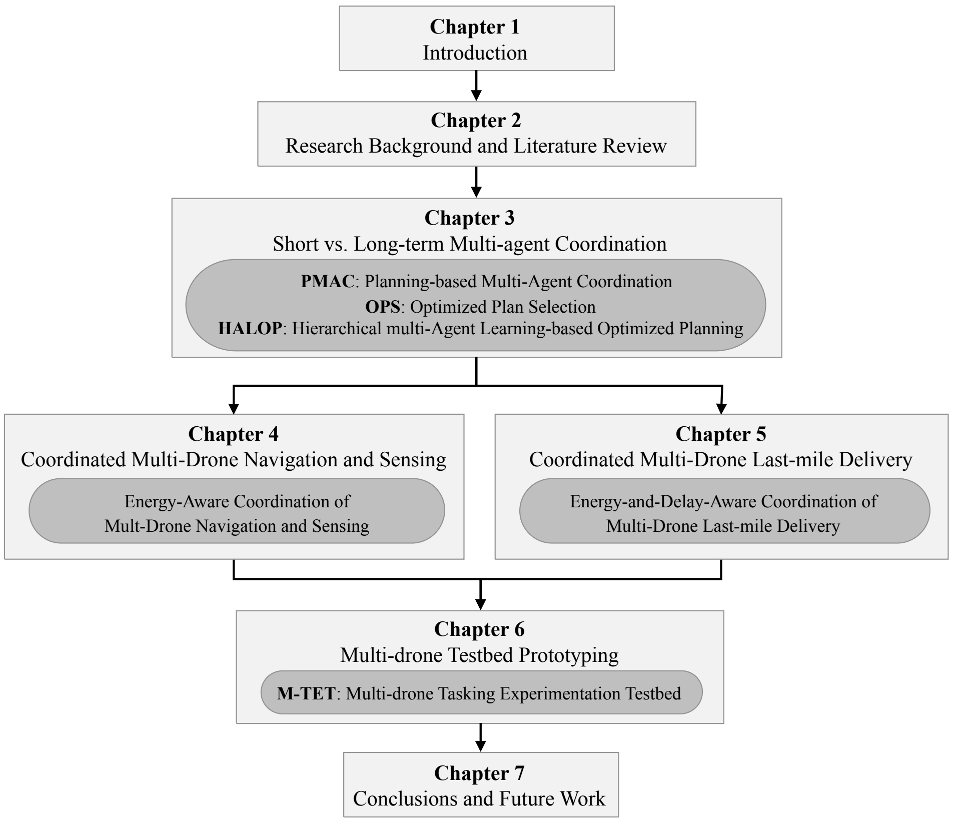

Chapter 1
Introduction
Chapter 2
Research Background and Literature Review
Chapter 3
Short vs. Long-term Multi-agent Coordination
PMAC: Planning-based Multi-Agent Coordination
OPS: Optimized Plan Selection
HALOP: Hierarchical multi-Agent Learning-based Optimized Planning
Chapter 4
Coordinated Multi-Drone Navigation and Sensing
Energy-Aware Coordination of Mult-Drone Navigation and Sensing
Chapter 5
Coordinated Multi-Drone Last-mile Delivery
Energy-and-Delay-Aware Coordination of Multi-Drone Last-mile Delivery
Chapter 6
Multi-drone Testbed Prototyping
M-TET: Multi-drone Tasking Experimentation Testbed
Chapter 7
Conclusions and Future Work

Figure 1.1: Graphical outline of this thesis.

for large-scale optimization within static environments, and the *Hierarchical multi-Agent Learning-based Optimized Planning* (*HALOP*) for strategic and long-term decision-making in evolving multi-agent systems.

Chapter 4 studies the applicability of PMAC in the urban sensing domain. A model is proposed to concentrate on optimizing the flight paths, data collection and recharging.

Chapter 5 studies the applicability of PMAC in the last-mile delivery domain. A model is proposed to account for both timely and multi-parcel route planning.

Chapter 6 illustrates the design, prototyping, testing and evaluation of the proposed testbed, named as *Multi-drone Tasking Experimentation Testbed* (M-TET).

Finally, Chapter 7 provides a summary, conclusions, and discussions which also highlights the possible future research directions.

Chapter 2

Research Background and Literature Review

This chapter presents a comprehensive review of the current research landscape, focusing on the problem, algorithms and application scenarios of distributed multi-drone coordination for task allocation. It is organized as follows: Section 2.1 outlines the fundamental nature of the multi-drone task allocation problem as well as hard environmental constraints. Section 2.2 explores distributed combinatorial optimization approaches to solve the task allocation problem, including distributed task allocation algorithms, bio-inspired algorithms, and other distributed heuristics. Section 2.3 focuses on dynamic, data-driven methods for task allocation in evolving environments by introducing state-of-the-art multi-agent reinforcement learning approaches and recent advances in solving scalability challenges. Finally, Section 2.4 contextualizes the discussed methods within real-world applications in intelligent transportation systems, particularly in urban sensing for traffic monitoring and last-mile delivery in logistics. Fig. 2.1 illustrates the taxonomy of combinatorial optimization approaches from literature review in the thesis.

2.1 Task Allocation and Constraints

Task allocation for drone swarms has become a critical area of research as swarms are increasingly deployed in missions such as environmental monitoring and target tracking [5, 37, 38]. Central to this challenge is the task allocation problem, which is commonly formulated as a multi-objective combinatorial optimization problem. This is a large class of problems [39] in which agents autonomously determine a number of finite options to choose from (operational flexibility). The agents may have their own

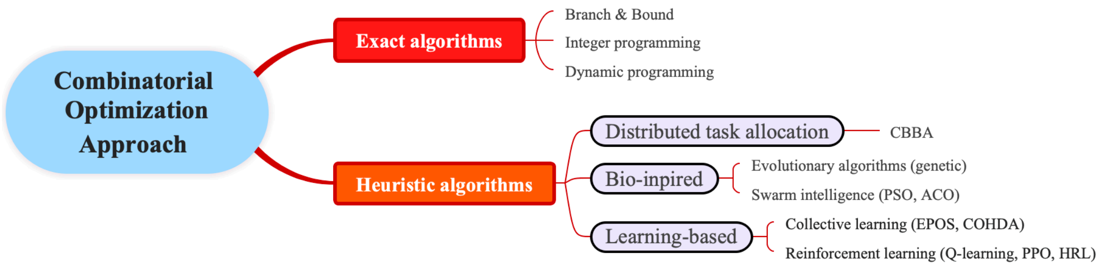

Combinatorial Optimization Approach
Exact algorithms
Branch & Bound
Integer programming
Dynamic programming
Heuristic algorithms
Distributed task allocation
CBBA
Bio-inpired
Evolutionary algorithms (genetic)
Swarm intelligence (PSO, ACO)
Learning-based
Collective learning (EPOS, COHDA)
Reinforcement learning (Q-learning, PPO, HRL)

Figure 2.1: The mind map with a taxonomy of combinatorial optimization approaches.

preferences over these alternatives, expressed with an individual cost for each option. However, such choices often turn out to be inter-dependent to minimize system-wide cost (non-linear cost functions) for which the agents may have (explicitly or implicitly) interest as well. These choices require coordination and computing the optimal combination of choices in an NP-hard problem [40].

In the context of multi-drone tasking, the goal is to assign tasks to multiple drones in a way that maximizes mission benefits, e.g., successful task completion, while minimizing associated costs, including travel distance, waiting time, and energy consumption [38]. Since these decisions are discrete and affect each other, the number of possible combinations increases rapidly, making the problem harder to solve, especially as the number of drones and tasks increases. This complexity highlights the need for efficient coordination strategies that can yield feasible and near-optimal solutions under real-world constraints [3].

The constraints on multi-drone task allocation typically include limited battery life, strict time windows for completing tasks, fly-restricted areas, collision avoidance requirements, and payload capacity restrictions [38]. These constraints are hard and must be strictly satisfied at all times to ensure the drones can complete their missions safely, efficiently, and without failure. To address these challenges, previous work often models the problem as a mixed-integer programming or mixed-integer linear programming problem [37, 41]. These formulations allow the representation of discrete decisions (such as task allocations and routing choices) alongside continuous variables (such as timing and energy consumption), enabling the enforcement of strict feasibility conditions through linear constraints. Solvers are then used to find optimal or near-optimal solutions that strictly satisfy all hard constraints. Specifically, earlier research [11, 42, 43] studies the impact of environmental obstacles and limited battery capacity on drone collaboration, presenting obstacle-aware and energy-efficient coordination models that aim to reduce mission completion time and improve task success rates. Theile *et al.* [44, 45] study the drone coverage path planning that accounts for varying power constraints, obstacle avoidance and no-fly zones.

Earlier research has employed exact algorithms to find globally optimal solutions for drone task allocation problems while satisfying hard and even critical operational constraints. Mailler et al. [46] introduce a constraint-based agent clustering approach, where a central controller uses a branch-and-bound strategy to search for feasible allocations. Thi *et al.* [41] enhance scalability and robustness by combining the cross-entropy method with a branch-and-bound algorithm to optimize drone-to-task allocations. Building on these efforts, recent studies have leveraged advanced optimization solvers that integrate multiple exact algorithms to handle nonlinear and large-scale problem instances more efficiently. For example, Pei *et al.* [6] propose a branch-and-cut algorithm for autonomous aerial urban delivery and implemented it using the Gurobi optimizer[1]. Song *et al.* [47] address the scheduling of drones with job splitting by formulating a mixed-integer programming model solved with CPLEX[2], significantly reducing computational complexity. Despite their accuracy, these centralized methods face critical limitations, including vulnerability to single-point failures, computational bottlenecks, and privacy risks when handling sensitive data such as location or health information. More importantly, their scalability is constrained, with time complexity growing rapidly as the number of drones and tasks increases.

2.2 Distributed Combinatorial Optimization

Distributed combinatorial optimization provides a framework for multiple agents or drones to make autonomous decisions while coordinating tasks efficiently across the system. By eliminating reliance on a centralized controller, distributed combinatorial optimization enables faster, more scalable responses to dynamic environments. This is particularly valuable in multi-drone task allocation problems, where drones must allocate and execute tasks collaboratively while adapting to real-time constraints and mission objectives. The rest of this section concludes common approaches to solve distributed combinatorial optimization problems:

[1]Available at: https://www.gurobi.com/
[2]Available at: https://www.ibm.com/products/ilog-cplex-optimization-studio

2.2.1 Distributed task allocation algorithms

Distributed task allocation algorithms mainly focus on market-based procedures, i.e., contract network algorithms and auction algorithms, including robust decentralized task assignment and auction-based task assignment [48, 49]. These algorithms provide a communication and negotiation mechanism through which agents or drones exchange information and coordinate task allocations based on well-defined consensus or bidding rules [50].

A notable example is the Consensus-Based Bundle Algorithm (CBBA), proposed by [51], which operates in two phases: a task inclusion phase where each drone greedily selects tasks to form a bundle, and a consensus phase, where conflicts among agents are resolved to ensure a consistent global allocation. Building on this framework, Fu *et al.* [52] extend CBBA by allowing drones with overlapping idle periods to negotiate and insert new tasks without disrupting previously assigned ones, thereby reducing overall path lengths. Wang *et al.* [53] further enhance CBBA with a two-tier bidding mechanism to maximize the number of tasks performed and minimize task execution time. More recently, Zhao *et al.* [54] propose the performance impact algorithm as an extension of CBBA by considering both the assignment cost of a task and its impact on the efficiency of already-allocated tasks. Qamar *et al.* [55] introduce the compromised dynamic performance impact algorithm to maximize task allocations and improve task prioritization in dynamic search-and-rescue scenarios. However, these approaches face challenges of high communication overhead and slow convergence, thus limiting their effectiveness in real-time scenarios.

2.2.2 Bio-inspired algorithms

Bio-inspired algorithms draw inspiration from the biological evolution and collective behaviors observed in nature [7]. In these approaches, a population of candidate solutions evolves or collaborates by sharing information to iteratively improve solution quality. Distributed variants of traditional bio-inspired algorithms, such as genetic algorithm [56], particle swarm optimization algorithm [57], ant colony optimization algorithm [58], have been widely adopted for distributed multi-agent coordination.

For instance, Wu *et al.* [59] propose a hybrid task allocation method that integrates a distributed genetic algorithm with an extended CBBA for time-critical tasks in dynamic disaster relief scenarios. Cao *et al.* [60] addressed dynamic battlefield task allocation with a mixed allocation method that divides the problem into group-level and member-level allocations, using enhanced particle swarm optimization and a distributed auction algorithm to reduce both time and communication complexity. Tang *et al.* [61] present a cooperative task reallocation algorithm for drones by combining fuzzy C-means clustering with ant colony optimization to maximize task completion rate and minimize flight distance. While these algorithms provide high flexibility and performance in complex scenarios, they often demand significant computational resources.

2.2.3 Other distributed heuristics

Other earlier work introduces the concept of *collective learning*, where participating agents autonomously select their local options, shaping their patterns of resource consumption and production. In this framework, learning is fully decentralized and emerges through the coordinated, remote interactions among agents. These interactions orchestrate collective decision-making over a communication network that is self-organized into a multi-level structure, typically forming a tree topology [17, 62].

In this highly challenging scope and problem setting, there is very limited work, mainly the COHDA [63] and EPOS [64, 65], that address a large spectrum of NP-hard combinatorial problems in the domains of Smart Grids and Smart Cities [64–67]. COHDA generalizes well in different communication structures among the agents that have full view of the systems, while EPOS focuses on hierarchical acyclic graphs such as self-organized trees to perform a cost-effective decision-making and aggregation of choices. As COHDA shares full information between agents, it has higher communication cost. The computational cost is lower at global level for COHDA compared to EPOS because of the agents' brute force search to aggregate choices. Both COHDA and EPOS focus on satisfying soft constraints, like minimizing cost functions that satisfy balancing (minimum variance and standard deviation) or matching (minimum root mean square error, residual sum of squares) objectives [65].

Furthermore, a recent study by Pournaras *et al.* [18] introduces the Iterative Economic Planning and Optimization Selections (I-EPOS), a decentralized coordination framework in which agents exchange information over a tree-structured communication topology. Each agent communicates only with its immediate parent and children, significantly reducing communication overhead compared to COHDA (full information exchange among agents) and lowering computational costs relative to the original EPOS (brute-force search over all local options) [18]. Follow-up research has demonstrated that I-EPOS is resilient by avoiding single points of failure and preserves agent privacy [68, 69], making it a practical solution for a wide range of real-world applications, including load balance of smart grids [65], electric vehicle charging [70], and house appliances electricity scheduling [71]. However, satisfying global hard constraints for these approaches is challenging as agents need to additionally coordinate for choices, whose potential violations are only confirmed at an aggregate level, which makes any rollback of choices to avoid violations particularly complex. In addition, these approaches are designed with fixed assumptions and limited foresight, making them suitable only for narrowly defined, short-duration tasks.

2.3 Learning-based Dynamic Task Allocation

Effective coordination among drones in dynamic environments requires the integration of real-time path planning and task allocation. To operate autonomously under uncertainty and evolving constraints, each drone must make local, distributed decisions that consider both immediate and future consequences. This calls for sequential decision-making that optimizes long-term cumulative rewards, a capability that traditional distributed combinatorial optimization algorithms often lack. While such algorithms excel at generating static or myopic solutions, they are generally not designed to learn or adapt over time based on environmental feedback.

2.3.1 Multi-agent reinforcement learning

To address these limitations, multi-agent reinforcement learning (MARL) offers a principled framework for enabling drones to make intelligent decisions through trial-and-error interaction with their environment [72, 73]. In MARL, each drone (agent) learns a policy, a mapping from observed states to actions, that aims to maximize cumulative reward over time. This process is grounded in the Bellman equation, which recursively defines the value function as the expected return from a given state, guiding agents toward optimal behaviors [72]. To improve performance, MARL methods balance exploration (trying new actions to discover better strategies) and exploitation (leveraging known actions with high reward). Depending on the learning objective, approaches may involve value-based methods (e.g., Q-learning), policy search methods (e.g., policy gradients), or combinations of both. Advanced algorithms such as actor-critic networks [74], deep deterministic policy gradient [75] and proximal policy optimization [76] have been successfully applied in continuous and dynamic multi-agent domains, allowing agents to learn in high-dimensional, partially observable environments with real-time adaptation capabilities.

These MARL methods are well-suited for drone coordination tasks where agents must make distributed decisions that account for long-term performance under changing conditions [8]. Dai *et al.* [77] explore multi-drone task allocation for exploration and destruction missions using three approaches: auction-based, vacancy chain, and deep Q-learning, aiming to minimize travel costs and improve task distribution. Liu *et al.* [78] introduce a multi-agent deep deterministic policy gradient algorithm for multi-drone task allocation, which adeptly adjusts to varying operation scales by distinguishing between local and global network inputs, effectively managing the complexities of changing drone cluster sizes.

2.3.2 Scalability challenges

Applying MARL to large-scale multi-drone systems requires overcoming the curse of dimensionality. In such settings, a large number of drones operate across wide areas, interacting and learning concurrently, which significantly increases the communication burden. In addition, when missions span long durations, drones must learn extended sequences of operations. These factors lead to an exponential growth in the joint state-action space, resulting in high computational and communication overheads, and reducing exploration efficiency.

To address this challenge, many studies adopt the centralized training with decentralized execution framework. It strikes a balance between two extremes: centralized training, where each agent can access to the true state of the environment and information of other agents via simulation, and decentralized execution, where agents chooses an action in accordance with locally partial information, helping to mitigate dimensionality [73]. In centralized training with decentralized execution, agents can share auxiliary information during training but act independently during execution, preserving decentralization.

Recent research also combines MARL with transfer learning [79], mean-field methods [80], and curriculum learning [81] to improve learning efficiency in multi-drone coordination. Transfer learning helps drones adapt faster by reusing knowledge from related tasks, as demonstrated by Venturini *et al.*[82], who reduce policy learning noise by filtering out irrelevant task knowledge. Mean-field method transforms the complex interaction within the population of agents into the simplified interaction between a single agent and the average effect from the overall population. This is validated by Chen *et al.*[83], who model energy-efficient coverage as a mean-field game and introduce a trust region optimization technique that balances communication efficiency with fairness and connectivity among drones. Xiao *et al.* [84] apply curriculum learning by breaking down a complex collaborative target-search task with sparse reward into sequential subtasks, including obstacle avoidance, exploration, and team coordination, and train agents in stages to gradually build up capabilities.

Another popular solution is Hierarchical Reinforcement Learning (HRL) [85], which is an important computational method aiming at large-scale problems. It structures decision-making into high- and low-level tasks, enabling agents to learn complex behaviors more efficiently by decomposing the problem. Unlike fixed abstractions, HRL dynamically learns these hierarchies, allowing for greater flexibility in responding to dynamic environments and achieving long-term goals [85]. In multi-agent systems, multiple HRL agents learn to coordinate their primitive action policies by optimizing a joint task objective and, occasionally, by sharing common information about their states and actions [85].

Existing work mainly train high-level and low-level policies simultaneously. Jendoubi *et al.* [86] leverage a high-level policy to plan the complex energy scheduling while using a low-level policy to handle the execution of specific actions associated with those plans, and both policies were updated based on the rewards received from the environment. Xu *et al.* [87] employ a dual coordination mechanism that facilitates the simultaneous learning of inter-level and inter-agent policies, addressing the instability arising from concurrent policy optimization. To facilitate exploration and hierarchical skill acquisition, an approach known as Hypothesis Proposal and Evaluation is created by Chuck *et al.* [88] and implemented in the software. The sample efficiency of this approach is derived from implicit presumptions of behavior in both the actual world and simulation environments. Wang *et al.* [89] pre-train low-level policy to ensure that a low-level policy learns effective primitive actions under macro action guidance. In the areas of multi-drone tasking, Liu *et al.* [90] combine a hierarchical framework with a

Figure 2.2: Photos of DJI UAVs (rotary).

model-free RL algorithm named Data-regularized Q [91], to improve the performance of drones in long-term autonomous navigation and obstacle avoidance. Nevertheless, training low-level policies in HRL is inherently non-stationary, as it depends heavily on the evolving outputs of high-level policies. Moreover, centralized training approaches often require access to sensitive agent-specific information, such as internal states and individual preferences, which may not always be desirable in decentralized systems.

2.4 Multi-drone Intelligent Transportation Systems

The modern transportation systems have integrated intelligent technologies and autonomous robotics to address the growing complexity of urban mobility, aiming to significantly enhance both the efficiency and safety of modern transportation systems [2]. Among the many autonomous platforms contributing to this transformation, drones play a pivotal role by elevating automation in two key ways: as aerial observers that offer a real-time “eye in the sky” to support road traffic operations, and as autonomous agents that carry out logistics or monitoring tasks independently [1, 92]. These capabilities position drone swarms revolutionize how transportation services are designed, delivered, and managed.

To support such applications, drone hardware choices and energy modeling are critical research concerns. This thesis mainly consider the multi-rotor drones, which utilize multiple rotors and propellers to generate the thrust required for vertical liftoff. These rotors produce upward forces that allow the drone to ascend, hover, and maintain stable flight. Unlike fixed-wing aircraft, multi-rotor drones are capable of vertical takeoff and landing and can hover in place with precision. These characteristics make them particularly well-suited for tasks that require stationary positioning or low-speed maneuverability, such as data collection and deliver parcels in urban environments. Lightweight commercial drones such as the DJI Phantom[1] and DJI Mavic[2] series are widely used for aerial video capture due to their affordability, portability, and high-resolution imaging. Meanwhile, larger platforms such as DJI FlyCart[3] enable heavy-payload logistics, making them suitable for long-range parcel delivery in a single flight (see Fig. 2.2).

A key engineering challenge in both of these cases is accurately estimating power consumption, which depends on physical factors including drone weight, airspeed, propeller number and length, and powertrain efficiency [34, 93, 94]. Understanding these characteristics is essential for mission planning, energy-aware coordination, and safe operation in real-world scenarios. Considering the flexibility of operations required in urban transportation scenarios, this thesis mainly considers the rotary-wing drones rather than fixed-wings [95].

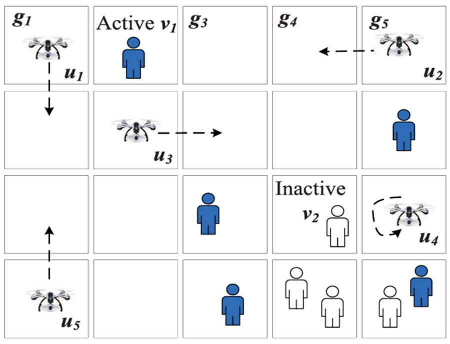
Building upon the foundational hardware and energy considerations, recent research focuses on task allocation algorithms that enable drone swarms to perform complex transport missions efficiently. This section conducts literature reviews that focus on urban sensing (e.g., traffic monitoring) and drone logistics (e.g., last-mile delivery).

2.4.1 Urban sensing and traffic monitoring

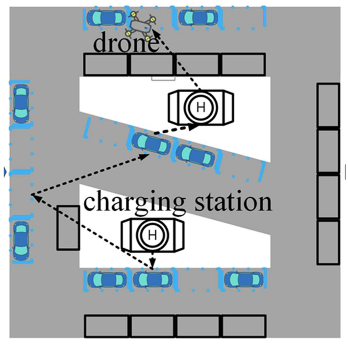
Nowadays, swarms of drones emerge in Smart Cites for spatio-temporal urban sensing [96–98]. Drones equipped with advanced sensors such as LiDAR and high-resolution cameras can form drone-assisted single-hop vehicular networks that generate and process time-sensitive data streams. For instance, swarms can capture images/videos of traffic-related information on public roadways; measure air temperature and humidity to support sustainable crop production; transmit real-time reports of natural disasters such as fire and car accidents; or accurately deliver goods in densely populated areas.

Beyond conventional performance metrics, e.g., throughput and latency, Samir *et al.* [101] introduce the concept of Age of Information and jointly optimize drone trajectories and scheduling policies to maintain timely data. Similarly, Zhou *et al.* [102] address joint route planning and task allocation in fixed-wing drone-aided mobile crowd sensing, emphasizing energy efficiency as a primary constraint. To overcome the inher-

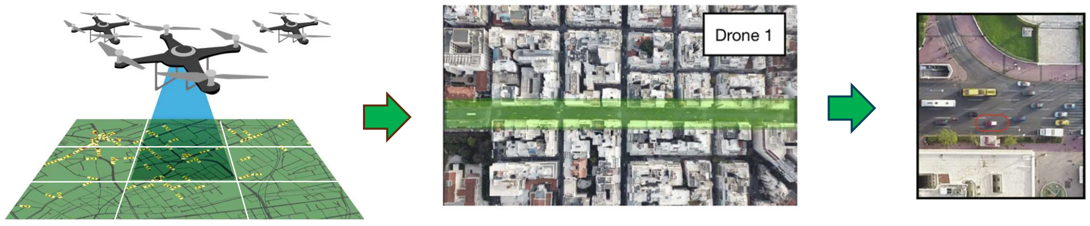
[1]https://www.dji.com/uk/products/phantom
[2]https://store.dji.com/uk/shop/mavic-series
[3]http://dji.com/uk/flycart-30



(a) Aerial- and ground-based mobile crowd sensing [99].



(b) Drone-based parking occupancy detection [100].



(c) Vehicle observation in drone-based traffic monitoring [4].

Figure 2.3: Scenarios of multi-drone urban sensing in related work.

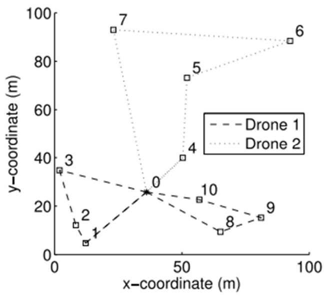
ent limitations of both human mobility and drone endurance, Ding *et al.* [99] propose a hybrid sensing framework that integrates ground-based mobile crowd sensing (e.g., humans with mobile devices) with aerial drone sensing (see Fig. 2.3(a)). Zhao *et al.* [100] introduce DroneSense, a smart scheduling algorithm that minimizes the number of sensing tasks by selecting key points of interest for drones to visit and inferring the state of unvisited locations, such as parking occupancy, while meeting overall sensing quality requirements (see Fig. 2.3(b)).

Among these urban sensing applications, traffic monitoring remains one of the most critical. Drone-based traffic sensing enables real-time vehicle detection, counting, tracking, speed estimation, and accident or congestion detection [103]. With their large field of view, rapid mobility, and ease of deployment, drones offer significant advantages over traditional static infrastructure in collecting traffic data, which further improves road safety and reduce operational costs [104]. For example, drones can be used for an accurate monitoring of traffic to detect traffic congestion at early stage. This allows traffic operators to apply mitigation actions that decrease the carbon footprint of a sector with one of the highest carbon emissions worldwide [4, 98]. Barmpounakis *et al.* [4] utilize a swarm of 10 drones to monitor traffic flows over several days in Athens' central business district, generating a rich dataset for analyzing complex urban traffic patterns (see Fig. 2.3(c)). Follow-up studies using this dataset have explored diverse phenomena, including the stopping behaviors of freight vehicles [105] and the relationship between lane-changing and lane choice [106].

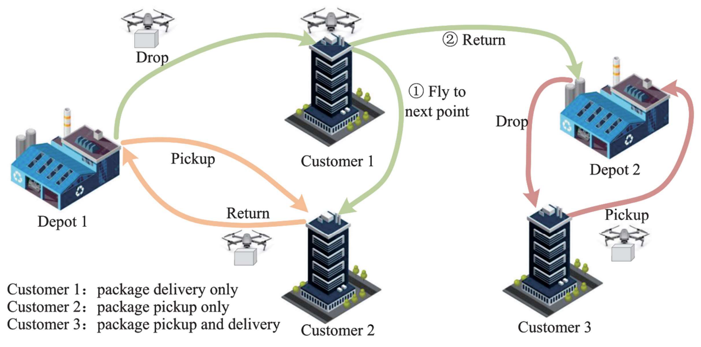
Task allocation and path planning further enhance traffic data collection. For example, Elloumi et al. [107] propose an adaptive drone-based system that adjusts drone trajectories based on real-time vehicle information. While effective in improving data acquisition, this method overlooks energy constraints and dynamic task requirements. Accurate vehicle observation is also essential for early detection of traffic congestion, enabling proactive interventions to reduce carbon emissions in one of the most polluting sectors. For instance, Bakirci et al. [108] apply the YOLOv8 algorithm to aerial images captured by custom drones, achieving high accuracy in vehicle detection. However, many of these studies face limitations in scalability, addressing only a small number of drones over short durations. Moreover, most lack validation on moving from simulation to outdoor live deployment, raising concerns about real-world applicability and robustness under practical deployment conditions.

2.4.2 Drone logistics and last-mile delivery

In the current landscape of logistics and supply chain management, profound technological advancements are notably marked by the distinguished use of drones [109]. In the past decade, pioneering companies such as Amazon, Domino's and Meituan have emerged as trailblazers in the practical application of drones for product delivery. For instance, the COVID-19 pandemic has highlighted the vulnerabilities of traditional delivery methods, as deliverymen risk spreading the virus. This was particularly problematic in quarantine zones, where customers faced difficulties in accessing logistics services [110, 111]. In contrast, drones offer a safer and more flexible alternative. Due to their high mobility, carrying capacity, and accurate GPS navigation, drones are able to deliver parcels directly to small places such as doorways and balconies, avoiding undesirable human contact. Furthermore, drones help alleviate the workload of ground vehicles, such as vans and trucks, thereby reducing road congestion. This shift from

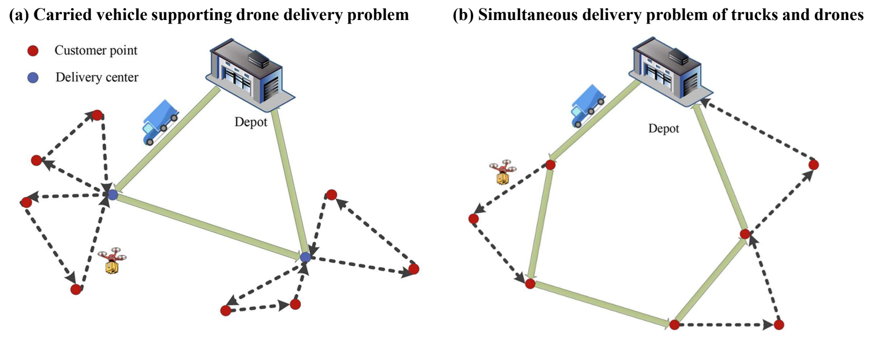


(a) Example of two drones flying three different routes [113].



Drop
② Return
① Fly to next point
Drop
Pickup
Customer 1
Depot 2
Depot 1
Return
Pickup
Customer 1: package delivery only
Customer 2: package pickup only
Customer 3: package pickup and delivery
Customer 2
Customer 3

(b) The drone package pickup and delivery mode and system [114].



(a) Carried vehicle supporting drone delivery problem
(b) Simultaneous delivery problem of trucks and drones
Customer point
Delivery center
Depot
Depot

(c) Schematic diagrams of truck-drone logistics distribution problems: [30].

Figure 2.4: Scenarios of multi-drone last-mile delivery in related work.

ground-based to aerial operations contributes to lower fuel consumption and emissions, supporting broader efforts toward achieving net zero targets [112].

Task allocation is vital to enhance the efficiency of drone swarms in completing delivery [113]. In coordinated delivery, effective task allocation guides swarms of drones to reach customers in the shortest possible time or distance. Prior works model this routing problem, which is essentially a variant of the traveling salesman problem and vehicle routing problem, as a mixed-integer programming problem and uses various heuristic algorithms to solve the complex combinatorial optimization [13]. The manually crafted heuristics approaches such as genetic algorithms, simulated annealing, and ant colony optimization have been widely applied [114–117], but can be time-consuming due to high searching space in environments. For example, when hundreds of customers need to be served by multiple drones, the number of possible delivery task allocations can

Table 2.1: Overall comparison in literature review.

Criteria **Ref.:**	[41, 46, 47]	[51–55]	[59–61]	[63–65]	[99–102, 118, 119]	[86–90]
Approaches	Exact algorithms	CBBA	Bio-inspired	Collective learning	MARL	HRL
Scalability	✗	✓	✓	✓	✗	✓
Adaptability	✗	✗	✗	✗	✓	✓
Decentralization	✗	✓	✓	✓	✗	✗
Energy-awareness	✗	✗	✗	✗	✗	✗
Hard constraints	✓	✗	✗	✗	✗	✗
Prototyping	✗	✗	✗	✗	✗	✗

grow exponentially, resulting in millions of combinations. This significantly increases the computational complexity of finding an optimal allocation.

More recently, reinforcement learning has emerged as an effective approach for solving last-mile delivery problems by learning optimal routing strategies without relying on manually crafted heuristics [118]. Tarhan *et al.*[119] extend reinforcement learning to multi-agent delivery scenarios and address scalability issues by employing state decomposition to reduce problem complexity, curriculum learning to guide exploration, and genetic algorithms to efficiently search the combinatorial space for drone-parcel assignments. Wu *et al.*[30] apply reinforcement learning combined with deep neural networks to jointly optimize the routes of drones and trucks. In this hybrid model, trucks act as mobile depots, transporting drones over long distances and deploying them for final deliveries, thus overcoming limitations in flight range and payload capacity of drones [120–122] (see Fig. 2.4(c)). However, these methods primarily focus on minimizing drone delivery time, without accounting for energy consumption. This is a critical limitation, as minimizing delivery time does not necessarily minimize energy use, especially given that drone power consumption is strongly influenced by payload weight [34]. In addition, existing methods often overlook delivery delays since a limited number of drones cannot serve many customers within a short time span. Strategically prioritizing a customer request with high delay is crucial in time-sensitive scenarios, such as medical supplies delivery during a pandemic [110, 123]. The gap between complex simulation and outdoor experimentation is still a critical concern.

2.5 Conclusions

In conclusion, this chapter presents the research background of multi-drone task allocation problems under various hard constraints, along with a review of solution approaches including exact algorithms, collective learning, reinforcement learning, and others, see Table. 2.1. These methods are examined in the context of key application domains such as urban sensing and logistics. The literature review highlights several important research gaps: First, there is a complementary gap between collective learning, which offers privacy preservation and efficiency for short-term task allocation, and multi-agent reinforcement learning, which enables long-term adaptability but struggles to scale to large problem instances and privacy issues. Second, energy-awareness remains underexplored, limiting the ability to accurately evaluate task performance and operational feasibility under real-world conditions. Third, a persistent gap exists between simulation-based validation and outdoor deployment, as many studies rely on over-simplified settings and lack testbed prototyping, hindering practical application of coordination strategies. Based on these findings, which align with the research questions defined in Chapter 1, the rest chapters of this thesis addresses these limitations and explore the proposed solutions in greater depth.

Chapter 3

Static vs. Evolving Multi-Agent Coordination

This chapter[1] introduces PMAC, the *Planning-based Multi-Agent Coordination* model. The PMAC is a generic and highly-efficient system model that can be used to solve the multi-objective combinatorial optimization problem, particularly those classified as NP-hard. The problem presents significant challenges in designing efficient algorithms for real-world applications of multi-drone task allocation [40, 124]. When agents have a set of self-determined options (or possible *plans*) to choose from, they dynamically adjust their decision alternatives in a distributed manner and optimize collective outcomes based on their behaviors for these plans. Each planned operation represents a sequence of tasks scheduled over a specific time period for execution. This planning-based model is particularly powerful because each plan encapsulates a full operation cycle, allowing agents/drones to reason at a higher level of abstraction. First, by having a complete overview of a potential trip, each drone can accurately estimate its energy consumption in advance, enabling more reliable and energy-aware decision-making. Second, since each discrete plan represents an extended operation, often lasting over 30 minutes, drones avoid making decisions at every minute or second. This significantly reduces the complexity of the decision space and allows the system to scale efficiently, even in large and dynamic environments.

However, in real-world scenarios, systems are constantly evolving, requiring agents to develop strategic foresight to achieve long-term optimization. Traditional algorithms (exact, heuristic and metaheuristic) for solving the combinatorial optimization problem are effective for immediate decision-making in the static environments, but often fail to account for future uncertainties and changing conditions [63, 125, 126]. To address

[1]This chapter is based on a published paper [40] and a paper on submission [16].

this limitation, the problem should be transformed into sequential decision-making problems, where agents may trade off short-term performance to maximize long-term benefits. Reinforcement learning has emerged as a powerful framework for modeling combinatorial optimization through dynamic programming [19, 127, 128]. By leveraging the Bellman equation, RL allows agents to evaluate the future impact of their current actions, guiding them toward high-quality approximate solutions (i.e., collective decisions) to NP-hard problems.

Furthermore, applying reinforcement learning techniques such as multi-agent reinforcement learning (MARL) to multi-objective combinatorial optimization problems remains open and pressing challenges: (1) As systems scale to multiple agents, the joint state-action space expands exponentially, making the training computationally expensive [129]. As a result, training requires more episodes and computational resources to explore and learn optimal policies; (2) The significant communication overhead forces agents to rely on partial observations and learn local parameter values, often prioritizing individual gains over system-wide efficiency [130]; (3) The centralized training paradigm in MARL requires access to all system-wide data, which may include personal and sensitive information, raising privacy concerns [131]. These long-standing challenges of scalability, efficiency, and decentralization highlight the limitations of MARL, while aligning with the core strengths of conventional algorithms to address the multi-objective combinatorial optimization problems, such as collective learning.

To address these challenges, the PMAC model firstly classifies static and evolving systems, and then introduces two combinatorial optimization approaches: the *Optimized Plan Selection* (*OPS*) method and the *Hierarchical multi-Agent Learning-based Optimized Planning* (*HALOP*) method:

Optimized Plan Selection. *OPS* targets at static environment by employing multi-agent collective learning. It offers a promising solution to coordinate agents to autonomously select a plan for task execution within a time period while preserving agents' privacy and autonomy [18, 39, 68]. To overcome the drawbacks of collective learning that address soft constraints only, a hard constraint satisfaction model is proposed. In addition, to satisfy the extremely strict (critical) constraints, *OPS* is flexible to choose another method for plan selection, such as exact algorithms (e.g., Branch-and-bound) [6].

Hierarchical Learning-based Optimized Planning. *HALOP* presents a hierarchical framework that integrates MARL and *OPS* across two layers, leveraging the strengths of both approaches. At the high-level layer, agents use MARL to learn strategic decision-making, such as selecting groups of plans and determining preferences or behaviors for plan selection. The specific plan selection is then delegated to *OPS* in the low-level layer. Based on the outcomes of this plan selection process, *HALOP* evaluates the long-term consequences of agents' collective actions through cumulative rewards, which are used to further update agents' high-level policies.

This chapter is organized as follows: Section 3.1 defines and formulates the multi-objective combinatorial optimization problem based on the task plans. Section 3.2 introduces the designed PMAC to address the problem. Section 3.3 and Section 3.4 illustrate the proposed approaches of *OPS* and *HALOP* in PMAC respectively. Section 3.5 experimentally evaluates the proposed approaches in a synthetic scenario. Section 3.6 compares the proposed approaches with related work. Section 3.7 discusses the new insights from the experimental results. Section 3.8 illustrates how the PMAC model apply to real-scenarios of multi-drone task allocation. Finally, Section 3.9 concludes this chapter.

3.1 Problem Description

The core problem is how to coordinate multiple agents to select their operations (i.e., decision-making) among alternatives (i.e., *possible operational plans*) such that they complete system-wide tasks efficiently over a long time span (e.g., an full-day mission).

Assume a set of software agents $\mathcal{U} \triangleq \{1, 2, ..., U\}$, i.e. a personal digital assistant. These agents perform complex tasks in a large-scale spatial model over a long time, which is denoted as $\mathcal{T} \triangleq \{1, 2, ..., T\}$. Each agent u has a number of K options to choose from at time t, $\forall u \in \mathcal{U}$, $\forall t \in \mathcal{T}$. Each option k is referred to as a possible plan, which is a sequence of real values (i.e., a vector) formulated as follows:

$$p^u_{kt} = (p^u_{ktd})^D_{d=1} \in P^u_t = (p^u_{kt})^K_{k=1}, \tag{3.1}$$

where d denotes the position index of a value inside a plan with the size of D.

Each agent selects one and only one plan, and thus the selected plan (i.e. the agent's choice) is defined as $p^u_{kt} \cdot x^u_{kt}$, where x^u_{kt} denotes the binary variable which takes 1 if the agent u selects the plan with index of k at time t, and 0 otherwise. All agents' choices aggregate element-wise to the collective choice, i.e., the *global plan* g_t of the multi-agent

Table 3.1: Mathematical notations used in Chapter 3.

Notation	**Explanation**
$u, U, \mathcal{U}$	Index of an agent/drone; total number of agents; set of agents
k, K, s	Index of a possible plan; total number of plans; index of selected plan
d, D	Position index of a value inside a plan; the plan size
p^u_k, P^u	The plan of agent u with index of k; the set of generated plans of u;
τ, g, C^u	The target of the system; the global plan; the environmental constraints on u
D, f_{D}	Discomfort cost of the plan p^u_k; inefficiency cost function
$f_{\text{P}}, f_{\text{I}}$	Plan generation function; inefficiency cost function
β	The parameters to make trade-offs between discomfort and inefficiency costs
l, L	Index of iterations in collective learning; total number of iterations
c, t_c	Index of a child of the agent; the branch response of child c
δ_c	Decision regarding whether the agent's child c should change its selected plan
$\mathcal{U}^{\text{P}}, \mathcal{L}^{\text{P}}$	Upper and lower bounds of the aggregated choices
$\mathcal{U}^{\text{D}}, \mathcal{L}^{\text{D}}$	Upper and lower bounds of the total discomfort cost of agents' choices
$t, T, \mathcal{T}$	Index of a time period; total number of periods; set of periods
S, A, R	Set of local states; actions; the reward function
σ_1, σ_2	Parameters to normalize discomfort and inefficiency cost in reward function
π, θ^π	The actor network, i.e., the policy function; the parameter of the actor network
Q, θ^Q	The critic network, i.e., the value function; the parameter of the critic network
h, H	Index of a sample in MARL buffer; total number of samples in a mini-batch from buffer
$\gamma, \mathcal{E}$	The discount factor in MARL loss function; Number of episodes
i, I, G_i	Index of a constraint of plans; total number of constraints; total number of plans in constraint i
W, C_{dnn}	Number of nodes per layer in neutral network; computational complexity of deep neutral networks

system, which is formulated as follows:

$$g_t = (g_{td})_{d=1}^{D} = \sum_{u=1}^{U} p_{kt} \cdot x^u_{kt}. \tag{3.2}$$

The plan of the agent u can be generated based on the current environment observed by u, formulated as follows:

$$p^u_{kt} = f_{\text{P}}(\tau_t, C^u_t), \tag{3.3}$$

where τ_t indicates the target tasks required for agents to complete, e.g., the required amount of data collected by drones; C^u_t denotes the environmental constraints on agent u, e.g., the flight range of drones. In each time period, the environment evolves such that the target tasks updates.

Agents' choices are made based on different, often opposing criterion. Each agent has its individual cost over the possible plans, measured by the *discomfort cost function* $f_{\mathrm{D}}(\cdot)$: It indicates the discomfort cost D^u_{kt} incurred by an agent executing a specific plan p^u_{kt}. It can be formulated as below:

$$f_{\mathrm{D}}(p^u_{kt}, x^u_{kt}) = \frac{\sum_{u=1}^{U}\sum_{k=1}^{K} D^u_{kt} \cdot x^u_{kt}}{U}. \tag{3.4}$$

Each agent can make independent choices to minimize their own discomfort cost. However, agents may also have interest to satisfy the general-purpose criterion: the *inefficiency cost*, which is formulated by the *inefficiency cost function* $f_{\mathrm{I}}(\cdot)$: Balancing (e.g. min variance) and matching objectives (e.g. min residual of sum squares) are examples for measuring inefficiency cost with a broad applicability in load-balancing application scenarios of drones, e.g., rerouting vehicles to avoid traffic jams. Since this cost function is non-linear, meaning the choices of the agents depend on each other, minimizing the inefficiency cost is a NP-hard combinatorial optimization problem [18].

To satisfying all of these (opposing) objectives, the problem is modeled as a mixed-integer programming problem, formulated as follows:

$$\min_{x^u_{kt}} \sum_{t=1}^{T} f_{\mathrm{D}}(p^u_{kt}, x^u_{kt}). \tag{3.5}$$

$$\min_{x^u_{kt}} \sum_{t=1}^{T} f_{\mathrm{I}}(p^u_{kt}, x^u_{kt}). \tag{3.6}$$

Subject to

$$\sum_{k=1}^{K} x^u_{kt} = 1, \forall u \in \mathcal{U}, \forall t \in \mathcal{T}. \tag{3.7}$$

$$\mathcal{L}^{\mathrm{P}} \leq \sum_{u=1}^{U}\sum_{k=1}^{K} p^u_{kt} \cdot x^u_{kt} \leq \mathcal{U}^{\mathrm{P}}, \forall t \in \mathcal{T}. \tag{3.8}$$

$$\mathcal{L}^{\mathrm{D}} \leq \sum_{u=1}^{U}\sum_{k=1}^{K} D^u_{kt} \cdot x^u_{kt} \leq \mathcal{U}^{\mathrm{D}}, \forall t \in \mathcal{T}. \tag{3.9}$$

The objective function (3.5) aims to minimize the *mean discomfort cost*, which denotes the time-accumulated average discomfort cost of selected plans per agent over all time periods. The objective function (3.6) aims to minimize the *inefficiency cost* of the system. Constraint (3.7) ensures that each agent must select exactly one plan from its generated options. Constraint (3.8) restricts the upper bound $\mathcal{U}^{\mathrm{P}}$ and lower bound $\mathcal{L}^{\mathrm{P}}$ of the aggregated choices of agents, i.e., the global plan. Constraint (3.9) restricts the upper bound $\mathcal{U}^{\mathrm{D}}$ and lower bound $\mathcal{L}^{\mathrm{D}}$ of the total discomfort cost of these choices.

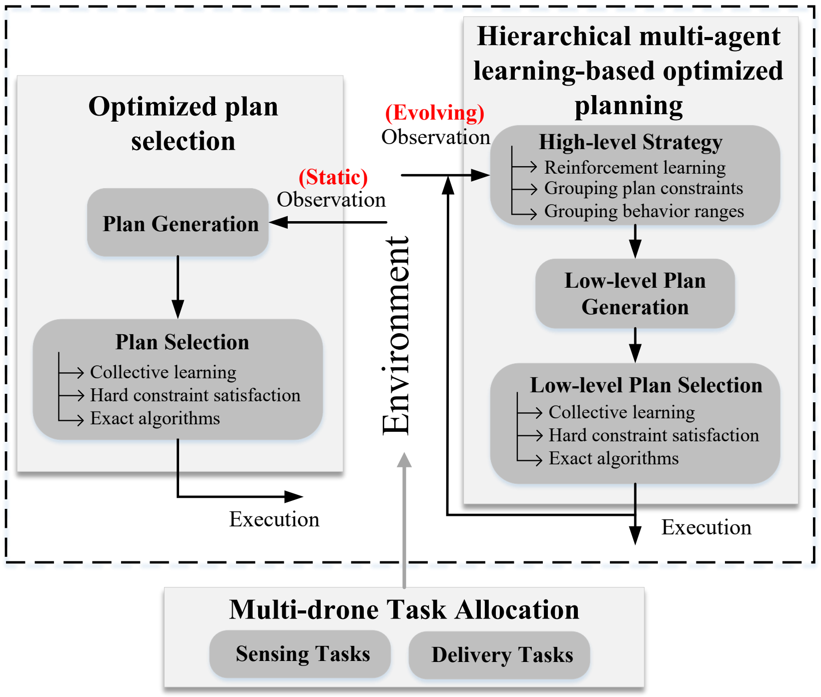


Figure 3.1: Framework overview of the designed PMAC model.

3.2 Framework Overview

To address the multi-objective combinatorial optimization problem, the *Planning-based Multi-Agent Coordination* (PMAC) model is proposed, as shown in Fig. 3.1. PMAC is a generic system model that contains two approaches: *Optimized Plan Selection* (*OPS*) approach and *Hierarchical multi-Agent Learning-based Optimized Planning* (*HALOP*) approach.

The *OPS* approach coordinates agents efficiently for task allocation in a distributed way within a given time window, i.e., the static environments. Each agent autonomously observes target tasks in the environment, generates plans, and make cooperative decisions through decision-making and choice aggregation. Once the plan is chosen, the agent will execute it and update the current environment. To assist agents select its optimized plan, this approach leverages collective learning for distributed optimization in a large-scale system. Moreover, a hard constraint satisfaction model is used to set strict constraints on collective learning. It also can choose the exact algorithms for global optimality under the critically hard constraints.

The *HALOP* approach, which is the core contribution of PMAC, extends the *OPS* approach to long-term optimization in the evolving systems. It leverages multi-agent reinforcement learning (MARL) to provide agents with strategic foresight in dynamic and evolving environments. It enables agents to evaluate cumulative rewards and anticipate the future consequences of their current actions, enhancing overall system efficiency and adaptability. Nevertheless, applying MARL to the problem introduces significant distributed coordination challenges: Explore a wide range of options and adjust decisions to effectively find an optimized one in the changing conditions, which increases the complexity of decision-making, leading to an expansive action and state space that makes training inefficient. Accordingly, the proposed model employs a hierarchical framework that leverages MARL to determine high-level strategies while offloading much of task selection burden to collective learning. Two types of high-level strategies are proposed, i.e., *grouping plan constraints* to reduce action space, and *grouping behavior ranges* to enhance Pareto optimality.

To apply the proposed approaches to the real scenarios of drones, PMAC defines the operations of multi-drone task allocation. It builds the environment where a swarm of intelligent and interactive drones perform various tasks, such as urban sensing data collection and last-mile delivery. The model will adopt an appropriate approach and an algorithm according to the specific conditions in these scenarios.

3.3 Optimized Plan Selection

This section introduces the key algorithm applied in the optimized plan selection (*OPS*) approach, i.e., a multi-agent collective learning algorithm as well as a hard constraint satisfaction model applied to the implementation of collective learning. In addition, the exact algorithms are considered as an option of plan selection in *OPS* for critical constraints.

3.3.1 Multi-agent collective learning

The *OPS* is implemented and integrated into multi-agent collective learning algorithm, named the Iterative Economic Planning and Optimized Selections (I-EPOS)[1] [18]. The

[1] Available at: https://github.com/epournaras/EPOS.

Algorithm 1: Optimized Plan Selection.

```
1  Input: Agent $u$, target $\tau_t$, algorithm type
2  Output: Selected plan $p^u_{kt}$ where $x^u_{kt} = 1$
3  Generate plans $(p^u_{kt})^K_{k=1}$ via Eq.(3.3)
4  if Algorithm type is collective learning then
5      for iteration := 1 to max-iteration-number do
6          Aggregate the plans of children and determine the decision $\delta^u$
7          Select a plan $p^u_{kt}$ via Eq.(3.10)
8          Share the aggregated plans and selected plan to parent
9          Wait for the decision and the global plan $g_t$ from parent
10         Send $\delta^u$ and $g_t$ to children
11     end
12 end
13 if Algorithm type is exact algorithms then
14     Send the $(p^u_{kt})^K_{k=1}$ to a third optimizer
15     Wait for the selected plan $p^u_{kt}$ from the optimizer
16 end
```

algorithm is selected because of its remarkable scalability (support a large number of agents), efficiency (low communication and computational cost) and resilience [18, 39]. The collective learning solves a large class of optimization problems in a decentralized manner, as formalized in the previous section. It is chosen due to its large spectrum of applicability in Smart City scenarios [64] as well as its superior performance in satisfying soft constraints [18].

Efficient coordinated choices are made using a self-organized [17] tree topology within which agents organize their interactions, information exchange and decision-making. Once agents have established their proximity-based relationship, e.g., Euclidean distance, they are positioned starting from the leaves up to the root, each interacting with its children and parent in a bottom-up and top-down [64, 132]. Fig. 3.2 illustrates an example of drones that self-organize into a tree communication structure for information exchange. The short/long-range communication between drones could rely on various methods, including WiFi, Bluetooth, cellular networks, and satellite link [14]. During the bottom-up phase, drone 3 aggregates the plans of its children, i.e., drone 1 and drone 2, and sends them to its parent agent with its own plan. During the top-down phase, each parent agent sends the global plans to its children such that all agents obtain the observed plan. As this distributed coordination among agents

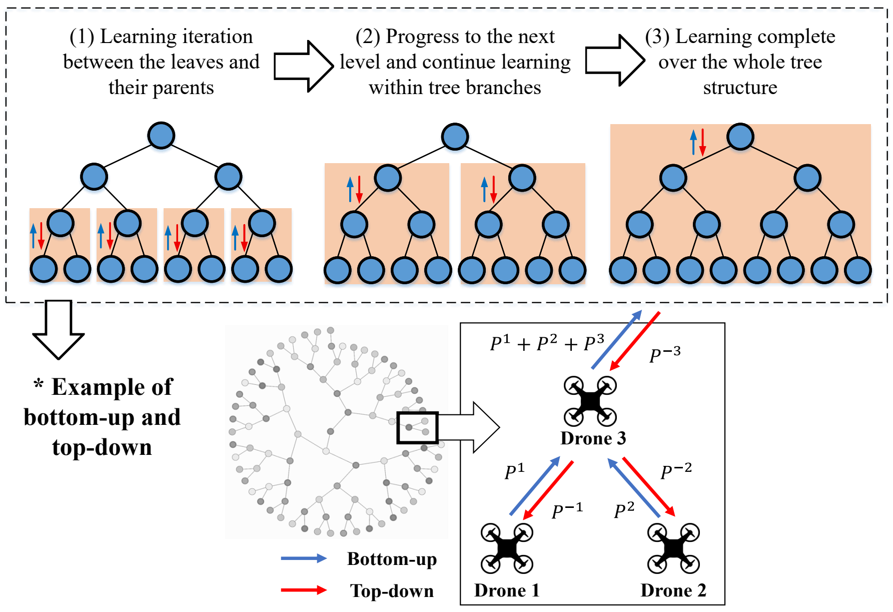

(1) Learning iteration between the leaves and their parents
(2) Progress to the next level and continue learning within tree branches
(3) Learning complete over the whole tree structure
* Example of bottom-up and top-down
$P^1 + P^2 + P^3$
P^{-3}
Drone 3
P^1
P^{-1}
P^2
P^{-2}
Drone 1
Drone 2
Bottom-up
Top-down

Figure 3.2: Bottom-up and top-down via a tree topology.

using hierarchical organization reduces the need for every node to communicate with every other node, the system is highly-efficient in collaboration and information exchange [18].

The whole iterative plan selection process is listed as follows (see algorithm 1):

During the bottom-up phase of the first iteration, each agent u (except for leaf nodes) observes the plans of its children and selects its own plan p^u_s. This selection depends on its selfish vs. altruistic behavior, e.g. whether they accept a plan with a bit higher discomfort cost to decrease inefficiency. Such multi-objective trade-offs are modeled with the parameter β^u such that:

$$p^u_{kt} = \underset{k=1}{\overset{K}{argmin}}\ (1-\beta^u_t)\cdot f_{\mathrm{I}}(p^u_{kt}, x^u_{kt}) + \beta^u_t \cdot f_{\mathrm{D}}(p^u_{kt}, x^u_{kt}), \tag{3.10}$$

where β^u_t represents the behavior of the agent u to balance both discomfort and inefficiency cost, $0 \leq \beta^u_t \leq 1$. As the value of β^u_t increases, the agent becomes increasingly more selfish, prioritizing plans with low discomfort cost at the expense of higher inefficiency cost. After plan selection, the agent combines the selected plan with the observed plans from its children into the branch response, and shares this combination to its parent node (if it is not the root).

During the top-down phase, however, each agent (except leaf nodes) sends the global plan g to its children. After the first iteration, each agent makes a choice taking into account the choices of all agents at a previous iteration, i.e., the global plan. During top-down, except for the updated global plan, each agent sends its child a decision δ_c regarding whether its child c should change its selected plan in the next iteration, which takes 1 if the agent changes its plan and 0 otherwise. This decision is determined by selecting a combination t_c, i.e., the tree branch response of a child c, that maximally improve the global plan. The tree branch grows as the learning progresses to the next level during the top-down phase, as shown in Fig. 3.2. The decision determination can be formulated as:

$$\underset{\delta_c}{min}\ f\left[g_l + \sum_c \delta_c \left(t_{c,l} - t_{c,l-1}\right)\right], \tag{3.11}$$

$$f(g) = (1-\beta_t^u) \cdot f_{\mathrm{I}}(p_{kt}^u, x_{kt}^u) + \beta_t^u \cdot f_{\mathrm{D}}(p_{kt}^u, x_{kt}^u), \tag{3.12}$$

where c denotes the index of a child for agent u, and l indicates the index of iterations in the collective learning.

The optimization in collective learning addresses the satisfaction of soft constraints, i.e., the objective function in Eq.(3.5) and Eq.(3.6), via various sequential information exchange, information aggregation and communication schemes that coordinate agents' choices. Furthermore, to consider the hard constraints on the aggregated choices and their costs defined in Eq.(3.8) and Eq.(3.9) respectively, the hard constraint satisfaction approach is required in the designed coordination model.

3.3.2 Hard constraint satisfaction

The collective learning for satisfying the hard constraints on aggregate choices and their costs is illustrated in this section.

The hard constraint satisfaction model ensures that aggregate choices and costs remain within defined upper and lower bounds while optimizing soft constraints [40]. Unlike traditional heuristics that focus on addressing soft constraints costs, which is the best effort to minimize all involved costs, this model directly addresses the challenge of hard constraint enforcement in multi-agent systems. To overcome the cold start problem, agents initially make decisions based on the highest likelihood of satisfying all hard constraints. Given the sensitivity of this process to self-organization, agents continuously reorganize after constraint violations until a stopping criterion is met. Once hard constraints are satisfied, agents transition to optimizing soft constraints while maintaining feasibility with minimal overhead [40].

Table 3.2: An example of applying collective learning and hard constraints with three agents, each with two plans.

Constraints	**Agent A**		**Agent B**		**Agent C**		**All Possible Global Responses**							
Agents' Plans (p)	**[3, 5]**	**[2, 7]**	[1, 3]	**[5, 2]**	**[6, 2]**	**[3,5]**	**[10,10**	**[14,9]**	**[7,13]**	**[11,12**	**[9,12]**	**[13,11**	**[6,15]**	**[10,14]**
1. Soft Constraints (Baseline)														
Minimize Inefficiency Cost	—3-5— = 2	—2-7— = 5	—1-3— = 2	—5-2— = 3	—6-2— = 4	—3-5— = 2								
Selected Plans	✓		✓		✓									
Selected Global Plan							✓							
2. Hard Constraints - $\mathcal{U}$ = [9,]														
Maximize Expected satisfaction	(9-3)+0=6	(9-2)+0=7	(9-2)+0=7	(9-5)+0=4	(9-6)+0=3	(9-3)+0=6								
Selected Plans		✓	✓			✓								
Selected Global Plan													✓	
3. Hard Constraints - $\mathcal{U}$ = [, 9]														
Maximize expected satisfaction	0+(9-5)=4	0+(9-7)=2	0+(9-3)=6	0+(9-2)=7	0+(9-2)=7	0+(9-5)=4								
Selected Plans	✓			✓	✓									
Selected Global Plan								✓						
4. Hard Constraints - $\mathcal{U}$ = [10,13]														
Maximize expected satisfaction	(10 - 3)+	(10 - 2)+	(10 - 1)+	(10 - 5)+	(10 - 6)+	(10 - 3)+								
	(13 - 5)=15	(13 - 7)=14	(13 - 3)=19	(13 - 2)=17	(13 -2)=15	(13 - 5)=15								
Selected Plans	✓		✓			✓								
Selected Global Plan									✓					
5. Hard Constraints - $\mathcal{U}$ = [9,9]														
Maximize expected satisfaction	(9 - 3)+	(9 - 2)+	(9 - 1)+	(9 - 5)+	(9 - 6)+	(9 - 3)+								
	(9 - 5)=10	(9 - 7)=9	(9 - 3)=14	(9 - 2)=11	(9 - 2)=10	(9 - 5)=10								
Selected Plans	✓		✓			✓								
Selected Global Plan									×					

Constraints on aggregated choices. For each element g_d of a global plan g, a hard constraint is defined by a range (envelope) of an upper $\mathcal{U}^{\mathrm{P}} = (\mathcal{U}^{\mathrm{P}}_d)^D_{d=1}$ and lower $\mathcal{L}^{\mathrm{P}} = (\mathcal{L}^{\mathrm{P}}_d)^D_{d=1}$ bound, where $\mathcal{U}^{\mathrm{P}}$, $\mathcal{L}^{\mathrm{P}}$ are also sequences of real values of equal size $|\mathcal{U}^{\mathrm{P}}| = |\mathcal{L}^{\mathrm{P}}| = |g_t| = D$. Each value d of the upper bound denotes that $\mathcal{U}^{\mathrm{P}}_d \geq g_{td}$, whereas for the lower bound it holds that $\mathcal{L}^{\mathrm{P}}_{td} \leq g_d$. The selected plan expected to satisfy all hard constraints at the initialization phase, during which the global plan g is not known, is estimated as follows:

$$p^u_{kt} = \underset{p^u_{kt} \in P^u_t}{argmax}\ \mathbb{E}(p^u_{kt}, x^u_{kt}, \mathcal{U}^{\mathrm{P}}, \mathcal{L}^{\mathrm{P}}), \tag{3.13}$$

where the expectation of satisfaction is given by:

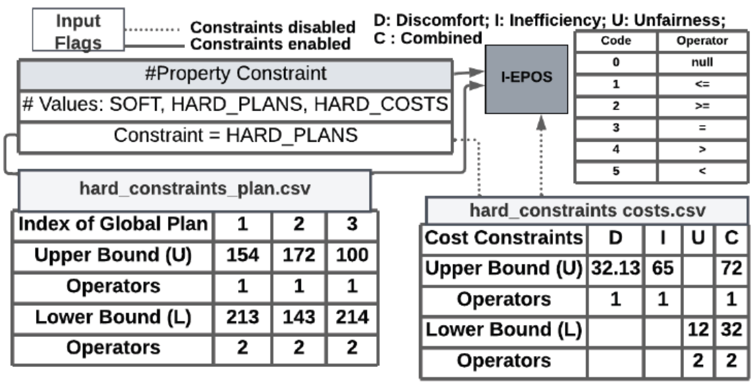
$$\mathbb{E}(p^u_{kt}, x^u_{kt}, \mathcal{U}^{\mathrm{P}}, \mathcal{L}^{\mathrm{P}}) = \sum_{d=1}^{D}(\mathcal{U}^{\mathrm{P}}_d - p^u_{ktd}) + \sum_{d=1}^{D}(p^u_{ktd} - \mathcal{L}^{\mathrm{P}}_d). \tag{3.14}$$

Constraints on aggregated costs. The modeling for the hard constraints on the aggregate costs is exactly the same as the one of the aggregate choices, where the expected satisfaction for each of the costs of $f_{\mathrm{D}}(p^u_{kt}, x^u_{kt})$ and $f_{\mathrm{I}}(\sum_{u=1}^{U} p^u_{kt} \cdot x^u_{kt})$ is calculated for upper and lower bounds with $|\mathcal{U}^{\mathrm{D}}| = |\mathcal{L}^{\mathrm{D}}| = 1$.

Constraint satisfaction rate. The effectiveness of the hard constraint satisfaction heuristic is measured by the satisfaction rate. This is the total number of satisfactions achieved out of a total number of trials made. These trials are often existing parameters of the optimization algorithms, for instance, random repetitions, or the order with which agents aggregate choices made to coordinate and optimize their own choices.

Table 3.2 illustrates an example of applying the heuristic in practice with three agents ($U = 3$), each with two plans ($K = 2, D = 2$). All combinations of possible plan selections make $2^3 = 8$ possible global plans. Hard constraints with an upper bound on the aggregate choices (global plan g) are introduced with an expected satisfaction of $\sum_{d=1}^{D}(\mathcal{U}^{\mathrm{P}}_d - p^u_{k,d})$. (1) The baseline scenario is the soft constraints that minimize the inefficiency cost $f_{\mathrm{I}}(g) \approx |p^u_{k,1} - p^u_{k,2}|$. The global plan $[10, 10]$ is the one with the inefficiency cost. (2) Agents choose plans that maximize the expected satisfaction. This results in the global plan of $[6, 15]$ that satisfies the hard constraint $\mathcal{U}^{\mathrm{P}} = [9,]$. (3) Similarly, the hard constraint $\mathcal{U}^{\mathrm{P}} = [9,]$ is satisfied with the global plan of $[14, 9]$. (4) Both new hard constraints $\mathcal{U}^{\mathrm{P}} = [10, 13]$ are satisfied with the global plan $[7, 13]$. (5) The second hard constraint $\mathcal{U}^{\mathrm{P}} = [9, 9]$ is violated by the selected global plan $[7, 13]$. $\surd$ indicates constraint satisfaction, and $\times$denotes constraint violation in p^u_k and g.

The decentralized hard constraint satisfaction model is implemented by filtering out


Input Flags
Constraints disabled
Constraints enabled
D: Discomfort; I: Inefficiency; U: Unfairness;
C : Combined
I-EPOS
#Property Constraint
# Values: SOFT, HARD_PLANS, HARD_COSTS
Constraint = HARD_PLANS
hard_constraints_plan.csv
hard_constraints costs.csv

Code	Operator
0	null
1	<=
2	>=
3	=
4	>
5	<

Index of Global Plan	1	2	3
Upper Bound (U)	154	172	100
Operators	1	1	1
Lower Bound (L)	213	143	214
Operators	2	2	2

Cost Constraints	D	I	U	C
Upper Bound (U)	32.13	65		72
Operators	1	1		1
Lower Bound (L)			12	32
Operators			2	2

Figure 3.3: Implementation of decentralized hard constraints satisfaction in collective learning.

the possible plans in Equation (3.10) that violate the given upper and lower bounds. However, in the first learning iteration, it is not possible determine these plans that violate the hard constraints with certainty because each agent only knows about the aggregate choices of the agents underneath (and not the ones above). As a result, the root agent may end up having no possible plan that does not violate the hard constraints. To prevent the likelihood of these violations, the agents make more conservative choices according to Equation (3.13) during the first iteration, aiming at maximizing the expected satisfaction of the hard constraints. Once the hard satisfactions are satisfied, the agents switch back to plan selection according to Equation (3.10), while keep filtering plans that violate the hard constraints. The agents cannot violate the hard constraints in these subsequent learning iterations because they always have the option to roll back to the choices made at the end of the first learning iteration during which hard constraints are satisfied (via the top-down phase).

Figure 3.3 illustrates the implementation of the hard constraints model on the open-source collective learning software artifact[1]. The implementation of the cost function interfaces is extended to filter out plans that violate the hard constraints, as well as the selection based on the expected satisfaction principle of Equation (3.13). The hard constraints are controlled via the main input parameter file of collective learning (Java Properties). Constraints on aggregate choices and costs can be activated and deactivated. Two input .csv files are introduced, one for each type of hard constraints.

[1]Available at https://doi.org/10.5281/zenodo.7791326

Both contain the upper/lower bounds and the coding of the operators.

3.3.3 Exact algorithms

Exact algorithms serve as an option for *OPS* to solve the combinatorial optimization problem under critical constraints. They are based on a complete and clever enumeration of the solutions space, which will eventually find the optimal solution. The well-designed exact algorithms can be used to obtain sub-optimal solutions by interrupting the search before its termination. This flexibility makes exact methods appealing and practical, and as such they constitute the core of modern optimization solvers as Gurobi, Cplex or Gecode. These optimizers efficiently solve the single-objective mixed-integer programming problem under constraints by combining various exact techniques, including branch-and-bound, cutting-plane, and etc. Therefore, the designed model is implemented using one of optimizers by setting the objectives Eq.(3.5) under the constraint (3.8), or setting the objective Eq.(3.6) under the constraint (3.9).

While these algorithms employ centralized coordination that depends on a third party to collect and process information, the decision-making itself remains distributed. Each agent independently generates local plans and shares only the resulting plans (not raw data) with a central entity. To facilitate this, similar to collective learning, agents self-organize into a tree topology for plan sharing, where the root agent aggregates the plans and sends them to a central server for computation. Therefore, the multi-agent system operates in a distributed manner, relying on the central server solely for computational support.

3.4 Hierarchical Learning-based Optimized Planning

The proposed method of *HALOP* is presented in this section. Compared to *OPS*, this approach integrates a state-of-the-art MARL to help each agent make a strategic decision-making for long-term optimization. It contains a two-stage hierarchical framework, where high-level strategic decisions guide low-level operational decisions.

According to Eq.(3.10) in *OPS*, agents' plan selection is determined by two factors: their local plans and behavior. Therefore, *HALOP* determines high-level action choices of the agents among (1) what alternatives they decide and (2) what behavior/preference they use to decide. Two types of strategies are employed respectively: *grouping plan*

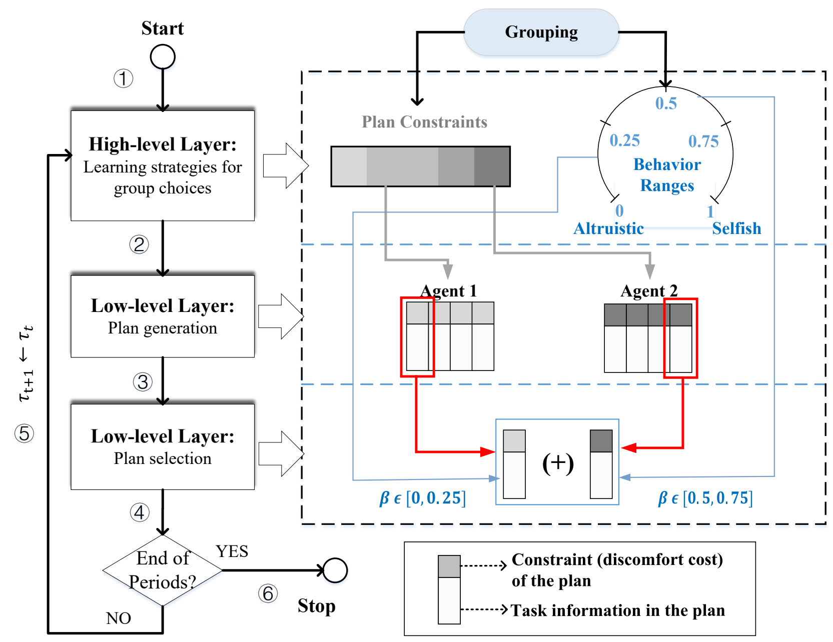

Start
①
High-level Layer: Learning strategies for group choices
②
Low-level Layer: Plan generation
③
Low-level Layer: Plan selection
④
End of Periods?
YES
NO
⑤
$\tau_{t+1} \leftarrow \tau_t$
⑥
Stop
Grouping
Plan Constraints
0.5
0.25
0.75
Behavior Ranges
0
1
Altruistic
Selfish
Agent 1
Agent 2
(+)
$\beta \epsilon [0, 0.25]$
$\beta \epsilon [0.5, 0.75]$
Constraint (discomfort cost) of the plan
Task information in the plan

Figure 3.4: The overall framework of the *HALOP* approach.

constraints and *grouping behavior ranges*. The strategy for grouping plan constraints divides the constraints of plans into different groups based on a criterion, e.g., the range of discomfort cost or the flight range of drones, and then chooses one of them. The strategy for grouping behavior ranges groups behavior β_t^u ranges and choose one to balance discomfort and inefficiency costs.

Fig. 3.4 illustrates the overall framework of *HALOP*. First, at a time period t, all agents construct into a tree topology where the root agent receives the target task from the server and then sends it to other agents via top-down phase. Next, each agent takes a high-level action, the results of choosing a plan constraint (grouping plan constraints) and a behavior range (grouping behavior ranges). Then, in the low-level layer, each agent generates plans via Eq.(3.3) based on the chosen constraint. Next, each agent coordinates to select its optimal plan based on the chosen behavior and obtains the global plan via collective learning or exact algorithms. Agents will execute their selected plans, which changes the environmental state and updates the target tasks in the next time period $\tau_{t+1} \leftarrow \tau_t$. Furthermore, the global plan is used for reward calculation calculation and state transition defined in Section 3.4.1. The state, action and reward of each agent are stored into a replay buffer in order to train the parameters of deep neutral networks in agents' policy, assisting them to explore optimized actions. More details are illustrated in Section 3.4.3.

3.4.1 Learning-based modeling

At each time period t, each agent observes the current target tasks and makes a decision on its plan, which further updates the target tasks in the next time period, influencing its new decision. Additionally, the decision-making of agents is not related to the target tasks before t, satisfying the Markov property [133]. Therefore, the above process is modeled as a Markov decision process. The components are listed as follows:

State. The state S_t^u includes the information observed by an agent u, such as the target τ_t, the selected plan p_{st}^u, the global plan g_t, and the discomfort costs of all selected plans $f_{\mathrm{D}}(\cdot)$.

Action. The action A_t^u of agent u at t is denoted as $A_t^u = (a_{imt}^u)_{i \leq I, m \leq M}$, where the agent u chooses a constraint index i from I constraints of plans, and a behavior range index m from M ranges (see Section 3.4.2). In addition, the state of agent u transitions from S_t^u to S_{t+1}^u after executing the action A_t^u and selecting a new plan p_{st}^u.

Reward. The expected immediate reward of an agent should be determined by specific task requirements (sensing or delivery). In this chapter, it is calculated based on the combined cost, indicating the sum of the normalized mean discomfort cost and inefficiency cost. According to the objective functions Eq.(3.5) and Eq.(3.6), the reward function at each time period t is expressed as:

$$R_t^u = -\sigma_1 \cdot \frac{1}{U} \cdot f_{\mathrm{D}}(p_{st}^u) - \sigma_2 \cdot f_{\mathrm{I}}(p_{st}^u). \tag{3.15}$$

where σ_1 and σ_2 are the parameters set by the system to normalize the discomfort and inefficiency cost respectively, $\sigma_1, \sigma_2 \in [0, 1]$.

3.4.2 High-level strategies

The high-level actions provide strategic guidance and constraints, shaping the discomfort cost of low-level actions to ensure that they align with overall objectives and efficiently achieve the optimization goals. The action consists of two strategies for group choices: the *grouping plan constraints* and the *grouping behavior ranges.*

Strategy for grouping plan constraints. The key idea of this strategy is to divide

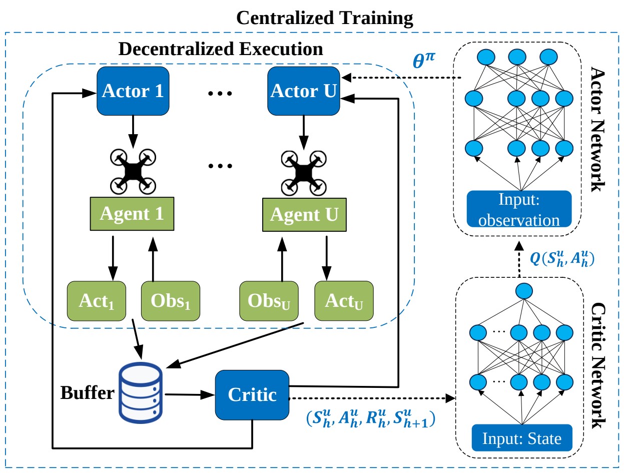


Figure 3.5: Centralized training and decentralized execution of *HALOP*.

the constraints of all possible plans into multiple groups based on a criterion, which could be the discomfort cost of plans, or the flight range of drones that perform tasking. By choosing one constraint, each agent generates the plans under the constraint and selects a plan from them. Since agents only takes action to choose a group, i.e., a constraint of plans, instead of selecting a plan from the entire set P_t^u, each agent significantly reduces the number of available action choices, thereby reducing the action space and computational cost. The selection of the constraints of plans can be defined as:

$$\{p_{ijt}^u, \forall j \leq G_i\} \leftarrow A_t^u \mid A_t^u = a_{imt}^u, \tag{3.16}$$

where i denotes the index of a constraint within a total of I constraints, j implies the index of a plan under the constraint i, and G_i is the total number of plans under constraint i, $G_i \leq K$.

Strategy for grouping behavior ranges. This strategy is designed to select from a group of behavior range rather than choosing among all possible values. The entire range $[0, 1]$ splits into M non-overlapping ranges of equal length, each range with an index of m. Then, each agent takes an action A_t^u to choose a range, and sets the mean value of the range as the agent's behavior β_t^u. This strategy is only feasible when the plan selection chooses collective learning. It can be formulated as follows:

$$\beta_t^u = \frac{m}{M} - \frac{1}{2M} \mid A_t^u = a_{imt}^u. \tag{3.17}$$

Algorithm 2: The training of *HALOP*.

```
Randomly initialize critic network $Q(\cdot)$, actor network $\pi(\cdot)$ with parameters $\theta^Q$, $\theta^\pi$
for episode := 1 to max-episode-number do
    Reset the target $\tau_t$ and the state of agents
    for period $t := 1$ to max-episode-length do
        for $\forall u \in \mathcal{U}$ do
            Take action: $A_t^u = \pi(S_t^u|\theta^\pi)$
            Grouping plan constraints and choose a constraint $C_t^u$ via Eq.(3.16)
            Grouping behavior ranges and choose a range via Eq.(3.17)
            Generate plans based on $\tau_t$ and $C_t^u$ via Eq.(3.3)
        end
        Coordinate agents to select plans through Algorithm 1
        for $\forall u \in \mathcal{U}$ do
            Obtain the next state and reward via Eq.(3.15)
            Store transition sample $(S_t^u, A_t^u, R_t^u, S_{t+1}^u)$ into buffer
            Sample a random mini-batch of $H$ samples from buffer
        end
    end
    Estimate advantage via Eq.(3.18)
    Calculate the probability ratio via Eq.(3.19)
    Update $\theta^\pi$ by minimizing the loss via Eq.(3.20) (3.21)
    Update $\theta^Q$ by minimizing the loss via Eq.(3.22)
end
```

3.4.3 Training and execution

The purpose of training of *HALOP* is to find the deep neutral network parameters of agents' policies that achieve stable and maximal reward during multiple episodes, each representing a whole process of plan generation and selection over all time periods. The well-trained policies are used as the execution of *HALOP*. As shown in Fig. 3.5, *HALOP* follows a centralized training and decentralized execution learning pattern. Agents can access the state, action and reward of other agents to train their policies efficiently, while operating independently during execution to apply to real-world conditions and ensuring decentralization. This is achieved by storing the experience transition $(S_t^u, A_t^u, R_t^u, S_{t+1}^u)$ of each agent into a central replay buffer. Every H episodes, a number of H transitions are randomly sampled from the buffer for training the deep neutral networks.

Moreover, *HALOP* employs actor-critic networks to combine the strengths of value-based methods (critic) and policy-based methods (actor) [134]. The actor learns a policy directly using policy gradient, enabling smooth and efficient optimal actions exploration, while the critic estimates value functions to guide and stabilize the updates of actor. There are two deep neural networks for each agent: (1) a centralized critic network $Q(\cdot)$, responsible for the quality (Q-value) of actions taken, and (2) multiple actor networks $\pi(\cdot)$, which represent the policy function that determines actions for agents. In addition, *HALOP* employs Proximal Policy Optimization (PPO) [135] to prevent detrimental updates of actor-critic networks and improving the stability of the learning process. The details of networks update are illustrated as follows:

The critic network estimates the reward associated with a transition using the Bellman equation. Its parameter θ^Q is updated by minimizing a loss function based on an advantage function $\hat{A}^u_h$. The advantage function is expressed as follows:

$$\hat{A}^u_h = R^u_h + \gamma \cdot Q(S^u_{h+1}, A^u_{h+1}) - Q(S^u_h, A^u_h), \tag{3.18}$$

where γ is a discount factor $0 < \gamma < 1$; S^u_h and A^u_h denote the state and the action of agent u in the sample h in a mini-batch from buffer, $h \leq H$.

Then, the critic network provides $\hat{A}^u_h$ to the actor network to increase the probability of actions that have a positive impact and decrease the ones that have negative impact. The actions are taken by drones as $A^u_t = \pi(S^u_t|\theta^\pi)$. To update the parameter θ^π of the actor network, PPO utilizes a policy ratio $\text{prob}(\theta^\pi, u)$, which is formulated as:

$$\text{prob}(\theta^\pi, u) = \frac{\pi_{\theta^\pi}(A^u_h|S^u_h)}{\pi_{old}(A^u_h|S^u_h)}, \tag{3.19}$$

where π_{old} denotes the older policy of the actor network in the previous iteration. This policy ratio is used to calculate the clip surrogate objective S^u_h:

$$C^u_h = \min[\text{prob}(\theta^\pi, u) \cdot \hat{A}^u_h,\ \text{clip}(\text{prob}(\theta^\pi, u), 1-\epsilon, 1+\epsilon) \cdot \hat{A}^u_h], \tag{3.20}$$

where ϵ is a hyperparameter; $\text{clip}(\cdot)$ indicates the clipping method to restrict the range of $\text{prob}(\theta^\pi, u)$ in order to preventing incentives from exceeding the interval $[1-\epsilon, 1+\epsilon]$. Finally, the parameters of both actor θ^π and critic θ^Q networks are updated by minimizing the loss functions $L_{\text{critic}}(\theta^Q)$ and $L_{\text{actor}}(\theta^\pi)$ respectively. They are expressed as follows:

$$L_{\text{actor}}(\theta^\pi) = \frac{1}{U \cdot H} \sum_{u=1}^{U} \sum_{h=1}^{H} C^u_h, \tag{3.21}$$

$$L_{\text{critic}}(\theta^Q) = \frac{1}{U \cdot H} \sum_{u=1}^{U} \sum_{h=1}^{H} (\hat{A}^u_h)^2, \tag{3.22}$$

In summary, the training process primarily involves the following steps (see Al-

Algorithm 3: The execution of *HALOP*.

```
1  Input: Agent $u$, target $\tau_0$, policy $\pi(\cdot)$
2  Output: Selected plans $(p^u_{s1}, ..., p^u_{sT})$,
3  for period $t := 1$ to max-episode-length do
4      Update the target $\tau_t$ and the state
5      Take action: $A^u_t = \pi(S^u_t|\theta^\pi)$
6      Grouping plan constraints and choose a constraint $C^u_t$ via Eq.(3.16)
7      Grouping behavior ranges and choose a range via Eq.(3.17)
8      Generate plans based on $\tau_t$ and $C^u_t$ via Eq.(3.3)
9      Select an optimized plan $p^u_{st}$ through algorithm 1
10 end
```

gorithm 2): It firstly performs the network initialization to set parameters in actor and critic networks within $[-0.1, 0.1]$ (Line 1). Next, in the exploration part (Lines 3-17) in each episode, each agent takes actions to choose a constraint of plans and a range of behavior. After coordination in low-level plan generation and plan selection, agents calculate their immediate rewards and transition to a new state. The buffer, a data storage structure used for experience replay, stores all the transitions of each agent (Line 14). For every H episodes, which equals to the batch size, H groups of transitions are randomly sampled from the buffer (Line 15). Finally, the algorithm updates the parameters in critic network θ^Q and actor network θ^π (Lines 18-21).

Moreover, the decentralized execution process is illustrated in Algorithm 3. With the input of initial target τ_0, the policy function $\pi(\cdot)$ and the high-level strategy, each agent runs the algorithm and outputs the selected plan p^u_{st} at each time period. Finally, each agent obtains its plans for all periods and execute them.

3.5 Experimental Evaluation

This section aims to validate the superior performance of the proposed approaches in the evolving systems and hard constraints. It firstly defines the experimental settings, including synthetic scenario, performance metrics, and baseline methods. Then, the performance comparison of different methods in the synthetic scenario is assessed.

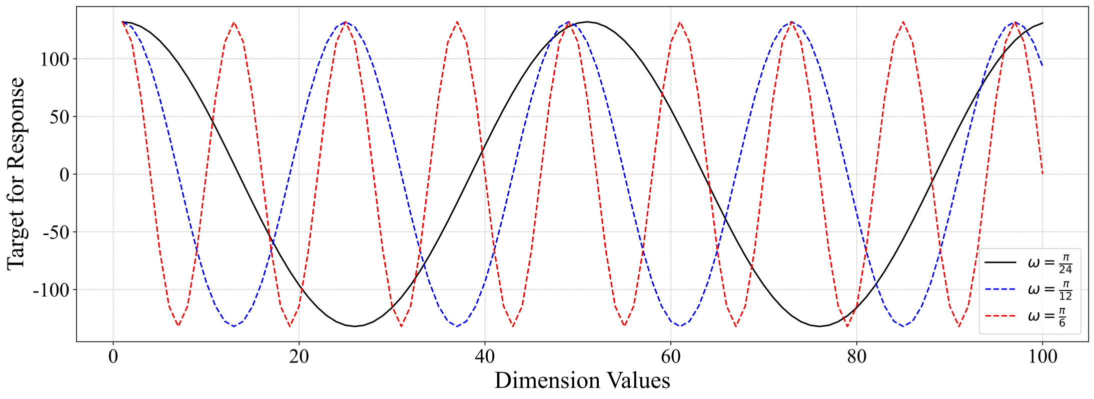


Figure 3.6: Synthetic target tasks for all agents of cosine waveforms by increasing the frequency multipliers ω.

3.5.1 Experimental settings

Scenario settings. The synthetic scenario is built by extracting data from the synthetic dataset [136]. It contains $1,000$ agents, each with 16 possible plans (i.e., vectors) that consists of a sequence of 100 values sampled from a Normal distribution centered at a mean of 0 with a variance of 1. The discomfort costs of plans for each agent are increasing linearly within the range $[0, 1]$. The plans are grouped using the quantile-based discretization function based on their discomfort cost. The goal of the optimization is to bring the global plan close to a target signal (a cosine waveform) over 16 time periods, while minimizing the discomfort cost of all agents. At each time period t, the update of the target can be formulated as: $\tau_{t+1} = \tau_t - g_t$, where τ_0 represents the target signal of the cosine waveform. Note that agents predict the future shape of waveform they construct (i.e., the global plan), which gradually approaches τ_0, updates the target, and thus the environment "seen" by each agent is evolving. Moreover, to increase the complexity of targets, the frequency multiplier of the cosine waveforms increases, as shown in Fig. 3.6. The cross-validation is applied: 80% plans of the datasets for training and 20% for testing. There are three types of scenarios for the experimental evaluation:

- *Basic synthetic scenario.* It is a benchmark with 40 agents, 16 time periods, 16 plans per agent and the target signal of $\omega = \pi/24$, aiming to compare the performance of different approaches.

- *Complex synthetic scenarios.* They are studied by varying the parameters to validate the scalability of the proposed solution. There are three dimensions: (1) the number of agents, (2) the number of plans (i.e., the plan volume owned by each agent), and (3) the complexity of target tasks (i.e., different frequency multipliers of cosine waveforms).

- *Hard constraint scenarios.* They are studied to test the hard constraint of the proposed solution. Three types of hard constraints are set to the aggregate choices (global plan g_t). In the basic synthetic scenario, hard constraints based on upper and lower bounds represent an envelope within which the global plan will not exceed the range. The agents are assumed here *altruistic*, $\beta_t^u = 0$.

Performance metrics. To evaluate the performance of both *OPS* and *HALOP*, two metrics are introduced: (1) Mean discomfort cost, which denotes the average of discomfort cost of all agents, defined in Eq.(3.5); (2) Inefficiency cost, which calculates the root mean square error between the target and global plan, defined in Eq.(3.6); (3) Combined cost, which measures the sum of the normalized mean discomfort cost and inefficiency cost. In simple words, the mean discomfort cost measures the cost of agents who execute the system tasks, while the inefficiency cost measures the overall quality of the executed system tasks. The combined cost provides a comprehensive assessment of the performance.

Baselines. A fair comparison of the proposed methods with related work is not straightforward as there is a very limited number of relevant decentralized algorithms [29, 86, 125]. Similar algorithms, such as particle swarm optimization, cannot be directly applicable to the optimization problem defined in Section 3.1. Those relevant algorithms (e.g., COHDA, CBBA) have higher computational and communication overhead than the collective learning used in *OPS* and *HALOP*, which is illustrated in Section 3.5.2. For this reason, this chapter focuses on ablation studies by testing standalone reinforcement learning, collective learning and exact algorithms. The baseline methods are introduced as follows:

- *MAPPO*: The name is Multi-Agent Proximal Policy Optimization, a state-of-the-art reinforcement learning technique [137]. It is adapted based on the designed model that each agent takes an action to choose one plan from the set P_t^u. Thus, its action space equals to the total number of plans generated by each agent $\sum_{i=1}^{I} G_i$. It employs the reward function of Eq.(3.15), the actor-critic networks and Proximal Policy Optimization. There is no high-level strategies for group choices but only address low-level plan selection.

- *HRL*: It is adapted based on the Hierarchical Reinforcement Learning [86] and designed model that each agent takes an action for high-level choices of plan constraints and then takes a second action to select a plan under the chosen constraint (i.e., low-level plan selection). The policy networks of both types of actions are trained independently by using two different actor networks.

- *System-optimal*: It is centralized method that aggregate the P_t^u from all agents and globally find optimally result of inefficiency cost through Gurobi optimizer [6]. As a type of algorithm in *OPS*, this method incorporates the plan generation and target update, but does not support any long-term strategic decision on task planning via learning methods. It only addresses low-level strategy without high-level strategies.

Furthermore, two variants of the proposed *HALOP* are considered: (1) *HALOP-P*, which uses the strategy for *grouping plan constraints* only and sets the agents' behaviors the same, and (2) *HALOP-B*, which uses the strategy for *grouping behavior ranges* only where agents select plans from the whole set P_t^u. As a comparison, *HALOP* takes both strategies together.

Algorithm settings. In this experiment, both *OPS* and *HALOP* leverage collective learning in the plan selection. The setting of algorithms contains both collective and reinforcement learning parts:

- During the coordinated plan selection via I-EPOS[1] in *OPS*, the communication structure among agents is set as a balanced binary tree [17]. In the one execution of I-EPOS, the agents perform 20 bottom-up and top-down learning iterations.

- For the reinforcement learning in *HALOP*, a total of $H = 64$ transitions are sampled as a batch in a replay buffer with a discount factor of $\gamma = 0.95$ and a clip interval hyperparameter of 0.2 for policy updating. The recurrent neural network is used with $W = 64$ neurons in the two hidden layers in both critic and actor networks. The activation function used for the networks is tanh. The models are trained over $\mathcal{E} = 2000$ episodes.

[1]I-EPOS is open-source and available at: https://github.com/epournaras/EPOS.

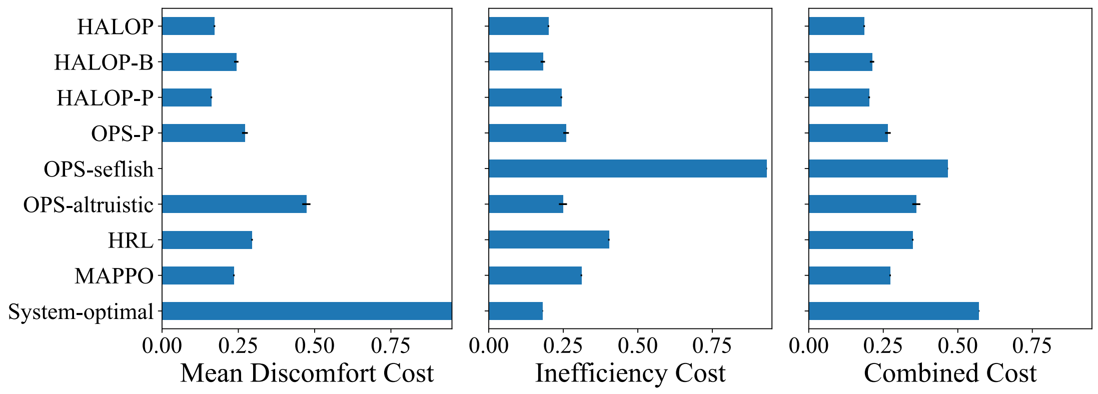


Figure 3.7: Cost comparison of all methods in the basic synthetic scenario (40 agents, 16 plans per agent and the target signal with $\omega = \pi/24$). The vertical lines denote the error.

In addition, several variants of *OPS* are also defined: (1) *OPS-altruistic* that agents behave altruistically ($\beta^u = 0$), (2) *OPS-selfish* that agents behave selfishly ($\beta^u = 1$), and (3) *OPS-P* that agents choose Pareto optimal point of behavior value. In terms of *HALOP*, several parameters are empirically selected due to the optimal performance: *HALOP-P* divides plans into 4 groups, each containing 4 plans; *HALOP-B* divides the behavior range into 4 ranges; and *HALOP* groups both 4 plans and 4 behavior ranges. The agent behavior is set as $\beta^u_t = 0.5$ in both *OPS* and *HALOP-P*. The σ_1 and σ_2 in Eq.(3.15) are set as 0.5 such that the reward is equivalent to the metric of combined cost defined in Section 3.5.1. More insights of the effect of different parameters are listed in Appendix A.1.

3.5.2 Evaluation on basic synthetic scenario

Fig. 3.7 illustrates the cost advantages of the proposed approach. *HALOP-P* achieves a combined cost that is 23.69% lower than *OPS-P*, benefiting from the use of deep neutral networks for function approximation. This assists *HALOP-P* to observe the environment and strategically select plan groups that reduce discomfort cost over time, while maintaining low inefficiency cost (The actions taken by agents per time are illustrated in Appendix A.1). Similarly, *HALOP-B* optimizes behavior ranges for agents by recommending them to behave selfishly at the beginning and then altruistically, minimizing combined costs. It also achieves the lowest inefficiency cost among all strategies,

Table 3.3: Comparison of computational and communication costs.

Approaches:	Computational Cost	Communication Cost
System-optimal [6]	$O(T \cdot K^U)$	$O(T \cdot K \cdot U)$
MAPPO [137]	$O(\mathcal{E} \cdot T \cdot U \cdot C_{\text{dnn}}(K))$	$O(\mathcal{E} \cdot T \cdot U^2)$
HRL [86]	$O(\mathcal{E} \cdot T \cdot U \cdot C_{\text{dnn}}(I + \frac{K}{I}))$	$O(\mathcal{E} \cdot T \cdot U^2)$
OPS	$O(T \cdot K \cdot L \log U)$	$O(T \cdot L \cdot \log U)$
HALOP-P	$O(\mathcal{E} \cdot T \cdot (U \cdot C_{\text{dnn}}(I) + \frac{K}{I} \cdot L \cdot \log U))$	$O(\mathcal{E} \cdot T \cdot L \cdot \log U)$
HALOP-B	$O(\mathcal{E} \cdot T \cdot (U \cdot C_{\text{dnn}}(M) + K \cdot L \cdot \log U))$	$O(\mathcal{E} \cdot T \cdot L \cdot \log U)$
HALOP	$O(\mathcal{E} \cdot T \cdot (U \cdot C_{\text{dnn}}(I \cdot M) + \frac{K}{I} \cdot L \cdot \log U))$	$O(\mathcal{E} \cdot T \cdot L \cdot \log U)$

only 0.49% higher than *System-optimal* that finds the optimality of inefficiency. However, *HALOP-B* tends to coordinate agents toward plans with higher discomfort cost, leading to a combined cost that is 4.81% higher than *HALOP-P*. By integrating the strengths of both high-level strategies, *HALOP* learns to optimize both plan groups and agent behaviors, achieving the lowest combined cost.

Moreover, with the help of effective coordination through collective learning, *HALOP* avoids unnecessary exploration, allowing it to more efficiently converge on the optimized global plan. As a result, it achieves 35.53% lower discomfort cost and 27.05% lower inefficiency cost compared to *MAPPO*. Note that the low-level policy on plan selection of *HRL* heavily depends on its high-level policy on plan constraints. The changing high-level policy leads to the plans selected by low-level policy are changing even taking the same action, which leading to non-stationary learning (see Appendix A.1) and 27.79% higher combined cost than *MAPPO*.

In addition, the complexity of all methods is compared. Given the number of nodes per layer in the deep neutral networks W, the state space $|S|$, and the action space $|A|$, the computational complexity of deep neutral networks is approximately $O(C_{\text{dnn}}(|A|)) = O(|A| \cdot W + W^2 + |S| \cdot W)$ [138]. The comparison of both computational and communication cost is shown in Table 3.3, where L denotes the number of iterations in *OPS*; $\mathcal{E}$ is the number of episodes. The results illustrate that the proposed method lowers computational cost by reducing the action space, especially dealing with a large number of plans K, outperforming *MAPPO* [137], *HRL* [86] and *System-optimal* [6]. *HALOP* also has lower communication cost as the number of agents increases, due to its efficient tree communication structure. More results of complexity comparison by increasing the number of plans and agents are shown in Appendix A.1.

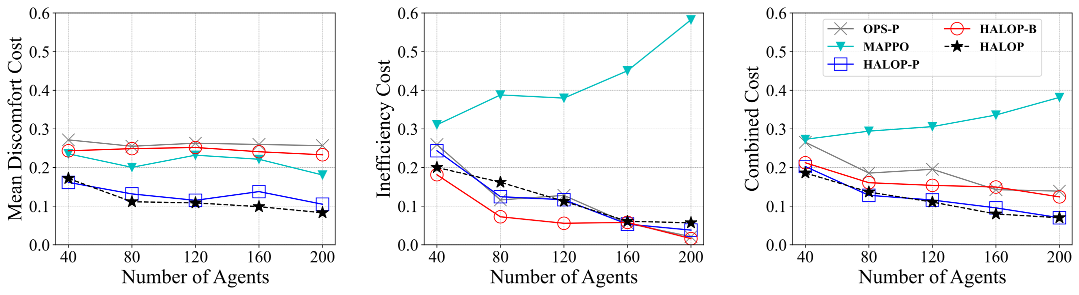


Figure 3.8: Changing the number of agents from 40 to 160 and fixing 16 plans per agent, 16 time periods and target complexity.

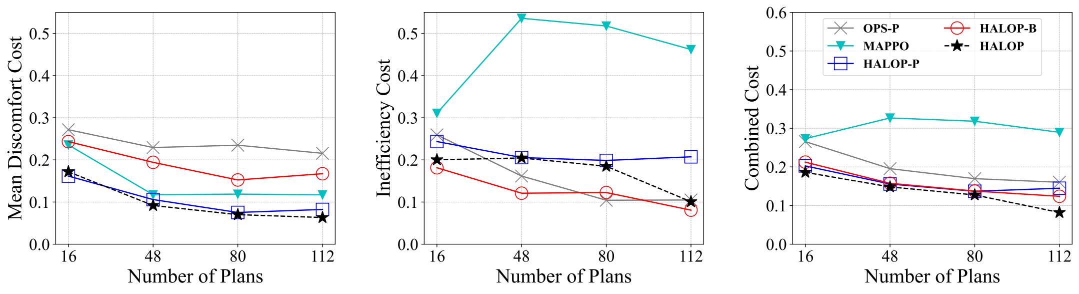


Figure 3.9: Changing the number of plans per agent from 16 to 112 and fixing 40 agents, 16 time periods and target complexity.

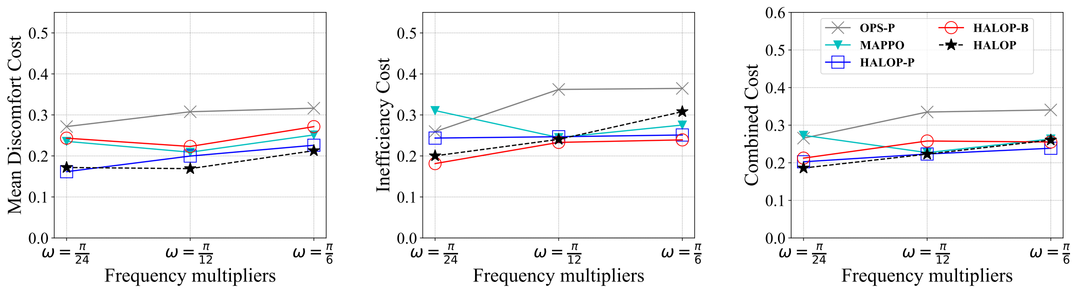


Figure 3.10: Changing the frequency multiple of cosine waves from $\pi/24$ to $\pi/6$ and fixing 40 agents, 16 plans per agent, and 16 time periods.

3.5.3 Evaluation on complex synthetic scenario

Based on the performance in basic synthetic scenario, this section compares the proposed approach with *OPS-P* and *MAPPO* in complex synthetic scenario by varying number of agents, number of plans and complexity of target tasks.

Number of agents. As shown in Fig. 3.8, if the number of agents increases from 40 to 200, there are more agents available to effectively coordinate and reach the target tasks. Therefore, the inefficiency cost of methods that use collective learning (*HALOP-P*, *HALOP-B* and *HALOP*) drops significantly by around 75%. In contrast, *MAPPO* has higher inefficiency cost with more agents due to the expanded state space, which hinders training efficiency. Moreover, *HALOP* strategically coordinates agents to choose plans with low discomfort cost, achieving approximately 24.34% lower combined cost than *OPS-P*. The results illustrate that *high number of agents significantly decreases the inefficiency cost of methods using collective learning.*

Number of plans. As shown in Fig. 3.9, if the number of plans per agent increases from 16 to 112, each agent has more options to respond to the complex environments. Thus, the combined costs of *HALOP* decreases by 55.91% as the number of plans increases, while still being lower than *OPS-P* by 48.75%. Unlike *MAPPO*, where agents must choose from a significantly larger action space, leading to increased training complexity, the action space of *HALOP* remains constant, contributing to its lower mean discomfort (45.94%) and inefficiency costs (64.26%). The results illustrate that *high number of plans decreases the discomfort and inefficiency cost of methods using collective learning.*

Complexity of target tasks. As shown in Fig. 3.10, if the frequency multiplier increases from $\pi/24$ to $\pi/6$ (see Fig. 3.6), agents struggle to achieve the collective goal. This leads to a linear increase of both mean discomfort and inefficiency cost in *OPS-P*. In contrast, *MAPPO* keeps the inefficiency cost relatively low and constant since it learns to effectively explore and select plans close to the cosine waveform with high frequency. Via reinforcement learning, the proposed *HALOP* achieves a combined cost that is 33.44% lower than *OPS-P*. However, the combined cost of *HALOP* exceeds *HALOP-P* at the complexity of $\pi/6$ as its exploration is restricted by the large action space. The results illustrate that *high complexity of target tasks increases the discomfort and inefficiency cost of all methods, but more slightly for those using MARL.*

3.5.4 Evaluation on hard constraint satisfaction

Fig. 3.11 shows that the aggregated plans for the soft constraint (*OPS*) along with three shapes of hard constraints (upper/lower bounds). Light-grey shaded areas represent

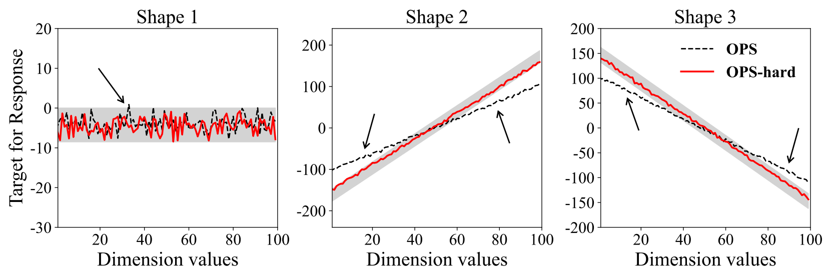
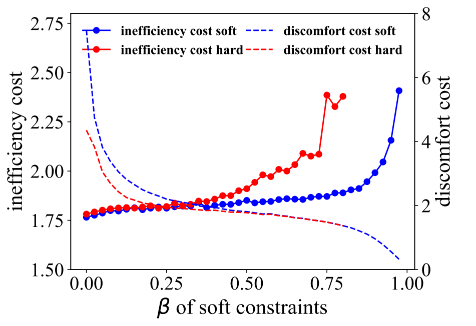


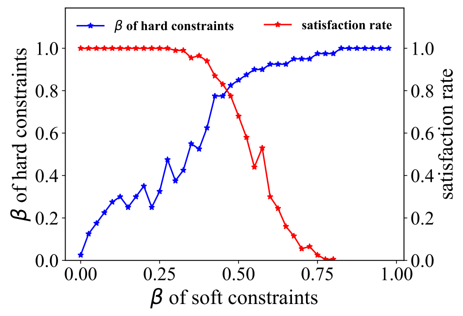
Figure 3.11: Optimization under soft and three types of hard constraints in the synthetic scenarios.

Table 3.4: Comparison of hard constraint satisfaction rate.

Synthetic scenario	*OPS*	*OPS-hard*	*System-optimal*
Shape 1	0.165	0.65	1.0
Shape 2	0.0	0.535	1.0
Shape 3	0.0	0.81	1.0



(a) β of soft constraints vs. inefficiency and discomfort costs.



(b) β of soft constraints vs. β hard constraints and satisfaction rate.

Figure 3.12: Required behavioral shift to mitigate the performance degrade of satisfying hard constraints.

the range between upper bound and lower bound. Arrows point to violations of hard constraints. Under soft constraints, the upper and lower bounds are violated, whereas the proposed method using hard constraints satisfaction model, named as *OPS-hard* prevent these violations. When the shape transforms from 1 to 2 and 3, constraints become stricter and prevent more violations in *OPS*.

Table 3.4 compares the satisfaction rates of *OPS*, *OPS-hard*, and *System-optimal*, based on 1000 simulation runs. The metric reflects how often each algorithm satisfies the hard constraints. The results show that while *OPS-hard* significantly improves constraint satisfaction over *OPS*, it still occasionally violates the constraints. In contrast, *System-optimal* consistently achieves full satisfaction of the hard constraints.

Satisfying hard constraints results in a degrade of the performance profile (lower inefficiency cost) achieved under soft constraints. The recovery from this degrade is measured here as the required social capital (behavioral shift) that agents need to offer such that soft and hard constraints have equivalent performance. The raise of social capital is measured by the reduction of the mean β_t^u value in the population of agents that makes them more altruistic, see Eq.(3.10). The following method is introduced to measure the behavioral shift: (1) Perform parameter sweep on I-EPOS under soft constraints for $\beta_t^u = 0$, to $\beta_t^u = 1, \forall iu \in \mathcal{U}$ with a step of 0.025. (2) For each I-EPOS execution in Step 1 with a β_t^u value, a discomfort cost and an inefficiency cost, run I-EPOS under a hard constraint on the mean discomfort cost with an upper bound value equals to discomfort cost (the one of the I-EPOS execution under soft constraints). (3) Derive the increased inefficiency cost under the hard constraint on the discomfort cost. (4) Find the β_t^u value from Step 1 that has the closest inefficiency cost with the one derived in Step 3 under the hard constraint. (5) Compare the two β_t^u values in Step 2 and 4. The difference represents the required mean behavioral shift to mitigate the performance degrade of hard constraints.

Figure. 3.12(a) illustrates the inefficiency cost and discomfort cost as a function of β under soft and hard constraints. It becomes apparent that hard constraints require a minimum and significant level of altruism, otherwise, inefficiency cost rapidly increases. Fig. 3.12(b) illustrates the required behavior shift to restore the performance loss as a result of satisfying hard constraints. The average satisfaction rate is 56.3%. For $\beta \leq 0.275$, the satisfaction rate is 100% for the synthetic scenario.

3.6 Comparison with Related Work

Reinforcement learning has emerged as a promising solution for addressing combinatorial optimization problems by modeling it as a Markov decision process [139]. Previous research has been focusing on applying reinforcement learning algorithms to approximate the solution to the NP-hard combinatorial optimization problems, including the traveling salesmen problem and knapsack problem [19, 127, 128]. However, to the best of our knowledge, there are very few works that study the problem by leveraging appropriate reinforcement learning techniques, such as multi-agent reinforcement learning. Therefore, this chapter compares the approaches of *OPS* and *HALOP* to related work in three design choices: (1) choice of collective learning, (2) choice of global information acquirement, and (3) choice of hierarchical framework.

Choice of collective learning. The choice of distributed optimization methods in the plan selection part requires to provide a scalable and efficient solution to the combinatorial optimization problem. Several earlier algorithms have demonstrated their optimization for multiple applications, for instance particle swarm optimization for vehicle path planning [125], ant colony optimization for routing in wireless sensor networks [126], and consensus-based bundle algorithm for risking task allocation [29]. However, these algorithms rely on frequent updates for paths or repeated combinatorial evaluations for tasks, which scales poorly with the number of agents and problem size. This is in contrast to the earlier work of I-EPOS [18, 65] that efficiently coordinates thousands of agents via a tree topology. Other highly efficient combinatorial optimization approaches using communication structure, such as COHDA [63] and H-DPOP [140], shares full information between agents, leading to higher communication cost than EPOS.

Choice of global information acquirement. It requires that every agent in reinforcement learning model can acquire the states and actions of all the other agents in the network. This can be impractical in large-scale problems, where sharing the state and action information may incur significant communication overhead. To address this limitation, prior work [130, 141] has explored deep reinforcement learning with partial observation, where agents rely on local estimates to approximate global rewards. Tilak *et al.* [142] model a distributed combinatorial optimization problem as a payoff game among agents and propose a partially decentralized reinforcement learning algorithm. Agents learn locally optimal parameter values, minimizing communication overhead while accelerating training convergence. Despite these improvements, partial observation limits an agent to acquire the full environment, making it struggling to plan ahead effectively. As a result, the agent tends to focus on immediate or local rewards, which may lead to decisions that are sub-optimal for the overall system performance in the long run. In contrast, the collective learning used in *OPS* and *HALOP* obtains global

Table 3.5: Comparison with related work in Chapter 3.

Criteria **Ref.:**	[125]	[29]	[18]	[142]	[86]	*OPS*	*HALOP*
Coordination for evolving systems	✗	✗	✗	✓	✓	✗	✓
Scalability with low complexity	✗	✗	✓	✓	✗	✓	✓
System-wide efficiency	✗	✗	✓	✗	✗	✓	✓
Autonomy and privacy-preserving	✗	✗	✓	✗	✗	✓	✓
Adaptability to diverse scenarios	✓	✓	✓	✗	✗	✓	✓

information for each agent with low communication cost.

Choice of hierarchical framework. The hierarchical framework is selected to divide complex tasks into high- and low-level subtasks, reducing the state-action space for each agent, inspired from hierarchical reinforcement learning (*HRL*) [85, 86]. Compared to existing action abstraction approaches that rely on predefined and offline abstractions, such as temporal abstraction [143], masking [144] and sequentialization [145], *HRL* provides flexibility in handling high- and low-level task structures and adapting to complex environments, as illustrated in Section 2.3. Nevertheless, the centralized training in *HRL* typically requires agents to coordinate and exchange personal information, which can inadvertently reveal their sensitive details such as the plan generation function and individual constraints. In contrast, the proposed approach respects the agents' autonomy to generate task plans, preserving their private information. Additionally, unlike *HRL*, the proposed approach allows greater flexibility in low-level plan selection by avoiding overly restrictive high-level constraints, resulting in improved overall system performance (see Section 3.5.2).

In summary, the research tackles the long-standing challenge of applying multi-agent reinforcement learning to combinatorial optimization problems, specifically addressing the scalability limitations in large-scale systems, the need for autonomy and privacy-preservation of agents, and the adaptability of solutions across diverse smart city scenarios. Table 3.5 illustrates the comparison to related work across multiple criteria (criteria covered ✓or not ✗). Here, *coordination for evolving systems* refers to the ability of the method to coordinate agents in the evolving environments; *scalability with low complexity* indicates the ability of the method to scale efficiently with low computational and communication overheads; *system-wide efficiency* checks whether the method has system-wide objective for all agents, whereas *autonomy and privacy-*

preserving checks the decentralization of the method that respect agent's autonomy and private information; and *adaptive to diverse scenarios* checks whether the method is generic to multiple real-world scenarios, rather than tailored to a specific one.

3.7 Discussion and New Insights

The experimental results demonstrate the superior performance of the designed PMAC. Several new scientific insights on the model are listed as follows:

Achieve a win-win synthesis of static and evolving optimization performance improvement in multi-agent systems. Unlike algorithms in *OPS* that is effective in static environment, *HALOP* in PMAC leverages deep neutral networks to effectively adapt to evolving environments and tackle complex tasks, leading to 23.69% lower system costs than *OPS* despite higher task complexity. Compared to traditional MARL methods such as *MAPPO*, *HALOP* enables agents to efficiently observe and aggregate information and maximize the system-wide performance by 31.29%.

Combine the strengths of both high-level strategies to improve scalability and Pareto optimality. The *grouping plan constraints* strategy in *HALOP* effectively reduces computational complexity by limiting the number of actions each agent must consider, ensuring scalable learning, faster training convergence and improved cost efficiency, even as the number of plans and agents increases. Meanwhile, the *grouping behavior ranges* strategy allows each agent to autonomously adjust its behavior within optimized ranges, significantly lowering both discomfort and inefficiency costs. This strategy is particularly effective with fewer plans and agents. By integrating these two complementary strategies, *HALOP* achieves a balance between scalability and solution quality, leading to more Pareto-efficient outcomes across diverse multi-agent scenarios. Moreover, it can switch to a single strategy in special cases, e.g., using *HALOP-P* for high complexity of target tasks and *HALOP-B* for low plan volume.

Ensure the individual privacy-preserving and system-wide resilience. Both *OPS* and *HALOP* in PMAC allow agents to self-determine their plan options without leaking their private information, e.g., the plan generation function and individual constraints. Each agent makes decisions locally, disclosing only minimal necessary aggregated information for coordination. Additionally, the system can dynamically reconfigure, repositioning drones within the tree in response to communication failures, ensuring that a single drone failure has minimal impact on information flow. The resilience of this model has been previously demonstrated in [132] with large-scale real-world datasets. Note that the critic network in *HALOP* is centralized but not privacy-intrusive without accessing to agents' private information.

Satisfy and mitigate hard constraints. The hard constraint satisfaction model assists the PMAC model to prevent violations of hard constraints to a very high extent. Even though for critical constraints, the PMAC model has flexibility to choose the exact algorithms (*System-optimal*) and achieve 100% satisfaction rate. Additionally, Results reveal the performance cost when hard constraints are introduced and how this cost can be mitigated by a behavioral shift towards a more altruistic behavior that sacrifices individual comfort for collective efficiency. These findings are invaluable for informing policy makers and systems operators of the required social capital that they need to raise to satisfy ambitious policies such as net-zero.

3.8 Operations of Multi-drone Task Allocation

This section defines the multi-objective combinatorial optimization problems for multi-drone task allocation, where a swarm of intelligent and interactive drones performs diverse mission operations. The primary objective is to complete sensing or delivery tasks while minimizing energy consumption associated with navigation and execution. Achieving this objective necessitates efficient task allocation and coordination. Therefore, the PMAC model is tailored to address the multi-drone task allocation problem.

3.8.1 Common models for drone-based task planning

Each drone in the swarm, controlled by a software agent, can generate multiple task plans via Eq.(3.3). A task plan represents a feasible route executed by a drone over a time period. For example, a drone may depart from a base station, sequentially visit several points of interest to perform missions (e.g., collect sensing data or deliver parcels), and return to the base. These multiple plan options reflect the operational flexibility and alternative choices available to each drone.

The energy consumed by a drone during task execution quantifies the discomfort cost of each plan. This is calculated using a power consumption model [34] that considers both flying and hovering power, based on physical parameters of the drone such

Table 3.6: Parameter comparison of multi-drone urban sensing and drone logistics.

Parameters	**Synthetic**	**Urban sensing (Traffic monitoring)**	**Drone logistics (Last-mile delivery)**
Plan type	Normal distribution	Navigation and hovering	Navigation
Discomfort	Plan index	Energy consumption	Energy consumption
System efficiency	Min inefficiency cost	Max mission efficiency Max sensing accuracy	Min expected delivery delay
Plan constraints grouping	Discomfort cost of plans	Flight range of drones	Flight range of drones
Spatial span	Large	Large	Large
Temporal span	Large	Large	Large
Environmental constraints	Soft & Hard	Soft & Hard	Hard

as body weight, battery weight, propeller length, ground speed and power efficiency. Note that the battery capacity imposes an upper bound on the total discomfort cost, as formulated in Eq.(3.9). More details on this power consumption model are provided in Section 4.3.

When the environments of multi-drone task allocation is evolving, drones should continuously determine and execute task plans over extended time periods that exceed their maximum flight time due to battery limitations. This requires drones to repeatedly cycle through navigation, task execution, and recharging by returning to a base station and departing again for subsequent tasks. Such iterative behavior allows the proposed learning methods in PMAC to operate effectively in large spatio-temporal environments. Furthermore, the selection of recharging places constraints the flight ranges of drones, which in turn affects their task performance. As a consequence, drones may depart from different locations at different times, necessitating a high-level strategy for *grouping plan constraints*. This grouping is based on navigation regions, defined by the flight range of drones, e.g., north, east, south, or west, rather than discomfort cost.

3.8.2 Comparison of sensing and delivery tasks

To demonstrate the applicability of PMAC, two practical scenarios of multi-drone task allocation are considered: urban sensing and drone logistics. Table 3.6 shows the parameter settings of these scenarios compared to the synthetic scenario.

Urban sensing and traffic monitoring. In this scenario, drones equipped with sensing modules are deployed to collect data across a spatial domain within a given time window. For instance, a swarm equipped with downward-facing cameras can capture real-time video for spatio-temporal traffic monitoring. In addition, drones may carry environmental sensors to collect temperature and humidity data in disaster-prone regions. A possible plan of a drone in this context defines which areas of interest it visits and how much sensing data it collects. The plan dimension d indicates the spatio-temporal reference (i.e., location and time), and the value p^u_{ktd} denotes the amount of data collected, equivalent to hovering time over the area of interest if the sensing frequency is fixed. The aggregated global plan reflects the total sensing output of the swarm, which must satisfy predefined data collection requirements, thereby maximizing both mission efficiency and sensing accuracy (The definitions are illustrated in Section 4.2).

Key aspects of this scenario include: (1) The deployment of a large number and heterogeneous drones for different sensing missions across the map; (2) Drones may belong to different entities or companies that prioritize privacy and are reluctant to share sensitive information; (3) The hard constraints apply, such as no-fly zones. These characteristics make collective learning an appropriate approach for plan selection, as it supports large-scale, decentralized, and privacy-preserving decision-making. Further details are elaborated in Chapter 4.

Drone logistics and last-mile delivery. In this scenario, drones equipped with carrying compartments or payload system perform parcel delivery from warehouses to customers. A possible plan of a drone specifies which customer locations it will serve. The plan dimension d indexes customer destinations, and the value p^u_{ktd} is a binary variable denoting whether a delivery is made or not. The global plan, formed by aggregating individual choices, defines the complete delivery schedule across all customers. This overall delivery aims to meet the customer requests on expected delivery time, i.e., minimizing the delivery delay. (The definitions are illustrated in Section 5.1)

Key considerations in this context include: (1) Logistics providers often aim to minimize operational costs by recycling a limited number of homogeneous drones; (2) Hard constraints are stringent, as each customer must be visited exactly once. These factors make exact optimization tools, such as Gurobi [6], particularly suitable due to their capability to handle problems with hard constraints efficiently. Further discussion is provided in Chapter 5.

3.9 Conclusions

In conclusion, the PMAC model for multi-objective combinatorial optimization problem is feasible by minimizing both individual agent costs and overall system-wide costs. Its approach of Optimized Plan Selection (*OPS*) effectively handle with different levels of constraints in the static environment by involving collective learning, hard constraint satisfaction model and exact algorithms. In the evolving multi-agent systems, however, the approach of Hierarchical multi-Agent Learning-based Optimized Planning (*HALOP*) integrates the decision-making of MARL with the coordination efficiency of short-term optimized plan selection. This synthesis outperforms standalone MARL and optimized plan selection method in terms of (1) discomfort and inefficiency cost minimization, and (2) low computational and communication overhead. The introduction of two high-level strategies for grouping plan constraints and behavior ranges further improves Pareto-optimal outcomes across varying scales of agent populations and plan complexities. The discussion on multi-drone task allocation provides a proof-of-concept for the broader applicability of PMAC.

The next two chapters introduce two complex scenarios of multi-drone task allocation whose design and implementation adopts the PMAC model.

Chapter 4

Coordinated Multi-Drone Navigation and Sensing

This chapter[1] mainly studies how swarms of drones self-assign their tasks of sensor data collection in areas of interest for spatio-temporal sensing. The sensing tasks includes various Smart City applications such as traffic monitoring (see Section 2.4.1). To assist swarms to complete sensing tasks efficiently, autonomous control of swarms and allocation of sensing tasks become a niche. Coordinated sensing involves the allocation of different sensing tasks to each drone while meeting the sensing requirements, drone capabilities and constraints [5]. Earlier work is proposed to address the task allocation problem for efficient and large-scale spatio-temporal urban sensing by swarms of drones [3, 146, 147]. Considering the heterogeneity and number of tasks, the problem is formulated as an NP-hard multi-objective combinatorial optimization problem to find the optimal allocation of sensing tasks. Task allocation algorithms designed to solve urban sensing problems range from market-based methods [52] to swarm intelligence [148]. To enhance robustness and minimize the effects of individual drone failures, distributed optimization approaches have been introduced. These approaches allow drones to autonomously self-organize and self-assign tasks in a decentralized manner, preserving the autonomy of individual drones.

However, mainstream approaches for drone sensing task allocations do not achieve scalability and long-term efficiency. This chapter discusses the two issues and corresponding solutions that the PMAC model illustrated in Chapter 3 can provide:

Distributed coordination at scale. To coordinate a swarm of drones at scale for distributed task allocation is a highly complex research endeavor. On the one hand, coordinating the sensing tasks of drones is complex, i.e., large areas of interest, with

[1]This chapter is based on a published paper [22] and a paper on submission [33].

varying sensing requirements and time-constrained missions. Certain areas with traffic jams or accidents may require for drones more fine-grained sensor measurements than areas with more regular traffic patterns. On the other hand, the inherent limitations in battery capacity influence spatio-temporal coverage. To tackle this complex task self-assignment problem, a distributed model is introduced, namely Energy-Aware Coordination of Multi-Drone Navigation and Sensing (EAC-MDNS) model. Using the proposed *OPS* approach, autonomous drones share information and allocate tasks cooperatively to meet complex sensing requirements while respecting battery constraints.

Slower is faster. Strategic foresight in navigation and sensing is crucial for optimizing drone operations in dynamic environments, significantly influenced by flying directions and future sensor data requirements. For example, drones aware of an expected increase in traffic can proactively fly to those areas, improving tasks such as vehicle detection, even if the short-term benefits are not evident. This “slower is faster” effect emphasizes the importance of long-term planning for overall effectiveness in drone sensing. Therefore, the proposed approach of *HALOP* can be used to address long-term optimization. It leverages multi-agent reinforcement learning (MARL) to determine the high-level strategies that select a group of navigation and sensing plans based on their spatial information (e.g., flying direction), while leaving the short-term plan selection to distributed optimization approaches. As a result, the high computation overhead and exploration inefficiency due to exponential state-action space in traditional MARL are overcome. This benefit becomes particularly significant when drones conduct urban sensing over an entire day, where efficient coordination of navigation, sensing, and recharging is essential.

This chapter is outlined as follows: Section 4.1 defines the sensing scenario by a swarm of drones and introduces the framework overview of EAC-MDNS model. Section 4.2 formulates the task self-assignment problem in the evolving multi-drone systems. Section 4.3 proposes a strategy of sensing plan generation based on a power consumption model[34] of drones, which will be used in the plan generation part of PMAC. Section 4.4 illustrates how the proposed *HALOP* apply to EAC-MDNS model. Section 4.5 evaluates the performance of both *OPS* and *HALOP* in short and long time periods respectively. Section 4.6 compared the proposed approaches with related methodologies that address urban sensing. Section 4.7 discusses the evaluation and outlines future work. Finally, Section 4.8 concludes this chapter.

Table 4.1: Mathematical notations used in Chapter 4.

Notation	Explanation
$u, U, \mathcal{U}$	Index of a drone; total number of drones; set of drones
$m, M, \mathcal{M}$	Index of a charging (base) station; total number of stations; set of stations
$n, N, \mathcal{N}$	Index of a grid cell; total number of grid cells; set of grid cells
$s, S, \mathcal{S}$	Index of a timeslot; total number of timeslots in a period; set of timeslots
$t, T, \mathcal{T}$	Index of a period; total number of periods; set of periods
a^u	The action or flying direction taken by u
P^u, P^{-u}	The plan of u; The observed plan of drones by u excluding P^u
$g_{ns} = \sum_{u=1}^{U} p_{ns}^u$	The aggregated plans of all drones at cell n and timeslot s
$R_n = \sum_{s=1}^{S} r_{ns}$	Required sensing value at cell n
$V_n^u = \sum_{s=1}^{S} v_{ns}^u$	Collected sensing value by u at cell n
τ, τ_{ns}	The target; target value at cell n and timeslot s
k, K	Index of a plan; total number of plans
$P^{\mathrm{f}}(u), t^{\mathrm{f}}$	Flying power consumption of u; flying time
$P^{\mathrm{h}}(u), t_u^{\mathrm{h}}$	Hovering power consumption of u; hovering time
C^u, e	Battery capacity of u; ; energy utilization ratio
K^u, J^u	Cell indexes within the searching range of u; Number of visited cells within K^u
β^u	Behavior of u in planning optimization
$\sigma_1, \sigma_2, \sigma_3$	Tradeoff parameters defined in the reward function
π, θ^π	The actor network, i.e., the policy function; the parameter of the actor network
Q, θ^Q	The critic network, i.e., the value function; the parameter of the critic network

4.1 System Model

This section defines the key concepts of sensing scenarios. Then, the EAC-MDNS model is introduced. Table 4.1 illustrates the list of mathematical notations used in this chapter.

4.1.1 Definitions and assumptions

Sensing map. Consider a swarm of drones $\mathcal{U} \triangleq \{1, 2, ..., U\}$ performing sensing missions, such as monitoring vehicles, over a grid that represents a 2D map. In this scenario, a set of grid cells (or points of interest) $\mathcal{N} \triangleq \{1, 2, ..., N\}$ are uniformly arranged to cover the map. The primary goal of the drones is to coordinate their visits to these cells to collect the required data. Furthermore, a set of base stations (or charging stations) $\mathcal{M} \triangleq \{1, 2, ..., M\}$, from which the drones depart from and return to, are located at fixed coordinates on the map.

Time periods and slots. A set of time periods is defined as $\mathcal{T} \triangleq \{1, 2, ..., T\}$. Each time period can be divided into a set of equal-length scheduling timeslots $\mathcal{S} \triangleq \{1, 2, ..., S\}$. In each timeslot, a drone can be controlled to fly to a cell and hover to collect sensor data.

Matrix of plans. To explain the short-term navigation and sensing of drones over the cells and timeslots in a period t, $t \in \mathcal{T}$, a possible plan of a drone u is introduced, denoted by $P_k^u(t)$, where k denotes the index of a plan in the generated plan set, $k \leq K$. $P_k^u(t)$ represents the specific navigation and sensing details, including the visited cells and corresponding energy consumption. The plan $P_k^u(t)$ is encoded by a matrix of size $N \times S$, with each element represented as $p_{kns}^u(t) \in \{0, 1\}$. Here, $p_{kns}^u(t) = 1$ denotes that the drone u hovers and collects all required data at cell n at timeslot s, whereas for $p_{kns}^u(t) = 0$ the drone does not hover at that cell at that time. Moreover, $P^{-u}(t) = \{p_{ns}^{-u}(t)|_{n\in\mathcal{N},s\in\mathcal{S}}\}$ denotes the plans observed by drone u in time period t, indicating that it incorporates and sums the selected plans of all other drones excluding its own. Based on the binary variable $x_k^u(t)$, which is similar to Section 3.1, the aggregated plans of all drones (i.e., the global plan) observed by u at cell n and timeslot s is formulated as follows:

$$g_{ns}(t) = p_{kns}^u(t) \cdot x_k^u(t) + p_{ns}^{-u}(t). \tag{4.1}$$

Matrix of required sensing values. In the context of a sensing task, each cell at a timeslot has specific sensing requirements that determine the data acquisition goal of drones. Such sensing requirements can be determined by city authorities as a continuous kernel density estimation, for example, monitoring cycling risk based on requirements calculated by past bike accident data and other information [149]. The high risk level of a cell at a timeslot represents the high importance of sensing (e.g., the crucial intersection of traffic flow), and thus a high number of required sensing values is set. The matrix of required sensing values at cell n is denoted as $R_n(t) = \sum_{s=1}^{S} r_{ns}(t)$, where $r_{ns}(t)$ denotes the required sensing value at cell n and timeslot s in the time period t. Based on actual sensing performance of drones according to Eq.(4.1), the sensing value collected by all drones observed by u at cell n is formulated as:

$$V_n^u = \sum_{s=1}^{S} v_{ns}^u = \sum_{s=1}^{S} p_{ns}^u(t) \cdot r_{ns}(t). \tag{4.2}$$

Matrix of target. In a real-world scenario, drones lack prior knowledge of the amount of required sensing values before they begin their sensing operations. Therefore, it becomes essential to build a target on the fly that instructs the drones regarding when and where they should or should not fly to given information from the environment. The matrix of the target is defined as $\tau(t)$, which denotes the sensing requirements for all drones at period t. The element of the target is denoted as $\tau_{ns}(t) = \{0, 1\}$, $n \in \mathcal{N}$, $s \in \mathcal{S}$, where $\tau_{ns}(t) = 1$ requires only a drone to visit the cell n at timeslot s, and $\tau_{ns}(t) = 0$ does not require sensor data collection by a drone.

Assumptions. Note that our model addresses task allocation problem for drone swarms, i.e., determining *what tasks to perform* and *where to sense*, rather than focusing solely on control and communication strategies that govern *how tasks are executed* (e.g., collision avoidance or minimizing latency). Therefore, to simplify the scenario, this paper makes the following assumptions: (1) Each drone is programmed to fly at a distinct altitude during its movement between cells to prevent mid-air collisions risks [150]. While previous work [151, 152] has explored a safe and cost-effective approach to a more realistic collision avoidance, this is not the primary focus of this chapter. (2) Each time period concludes a flying period and a charging period: the time that drones perform sensing, and the time for charging. All drones finish charging before the next time period begins. This chapter assumes that each charging period is of equal duration and provides sufficient time for the drones to be fully charged. (3) Each charging station is adequately equipped with charging capacity, enabling multiple drones to charge simultaneously without the need for queuing. This arrangement prevents any delays in the charging process. (4) Drone computations are offloaded to a remote edge-to-cloud infrastructure, where decisions are made based on data reliably transmitted from the drones via stable communication links [153, 154].

4.1.2 System overview

As shown in Fig. 4.1, a swarm of drones that perform sensing over a grid-a 2D map over the spatial illustrative model. A dispatch u is defined as a sensing task between the departure and return of a drone. Drones are equally distributed over these base stations, and have to return to the original base stations from which they depart.

Sensing map requirements. In the context of a sensing task, each cell n at a time period t has specific sensing requirements that determine the hovering duration and data acquisition of drones. For example, see Fig. 4.1. It is assumed that the cell A

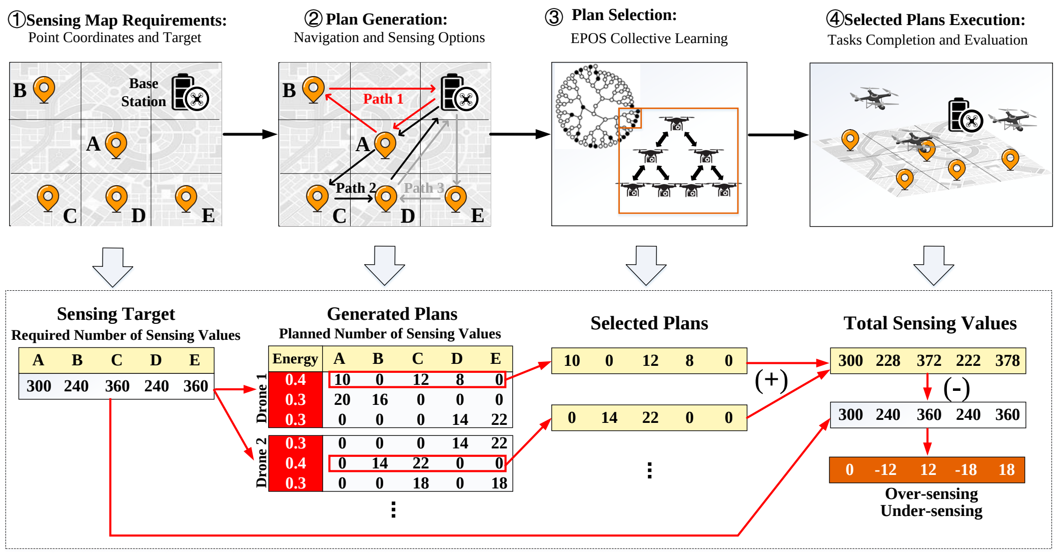


Figure 4.1: Framework overview of the EAC-MDNS model.

requires the total sensing values of 300 at a time period. The sensing requirements of cells at a time period are set as the *targets* encoded by a vector of size N, e.g., $\{300, 240, 360, 240, 360\}$.

Plan generation. To coordinate the allocation of sensing tasks, the drones use the proposed *OPS*. Given the sensing map, each drone, controlled by a local software agent, autonomously generates a finite number of discrete navigation and sensing alternatives, which provide flexibility for the drones to choose in a coordinated way. Each alternative has sensing details as well as an estimated energy consumption; the alternatives are referred to as the *possible plans* and each comes with a (normalized) *cost* respectively. For instance, a plan encoding that a drone travels and hovers over the cells A, C and D to collect respectively 12, 10 and 10 sensing values over a time period is encoded with: $\{12, 0, 10, 10, 0\}$ with $N = 5$. The energy consumed by a drone that carries out its planned tasks is calculated via a power consumption model [34] with input the specification of the drone (weight, propeller, and battery parameters). This model can estimate the cost of navigation and sensing plans, and emulates the outdoor environments [26].

Plan selection and execution. To make coordinated plans selection, the agents of the drones connect into a tree communication structure within which each interacts with its children and parent in a bottom-up and top-down fashion to improve plan selections iteratively [18]. The objective of this method is to select the optimal plan for each agent such that all choices together add up to maximize the sensing quality: the overall sensing data collected by all agents matches well the required data (target, see Fig. 4.1). In contrast, the sensing accuracy is a result of *over-sensing* and *under-sensing*. For instance, in Fig. 4.1, the sensing value of 372 for which drones hover over the cell C exceeds the requirements of 360 (over-sensing), whereas drones hover over the cell E to collect 222 sensor values that is lower than the requirements of 240 (under-sensing).

4.2 Problem Statement and Formulation

This chapter aims to solve the task self-assignment problem for urban sensing in the evolving multi-drone systems. Based on the defined scenarios settings and assumptions, the problem can be stated as: *To find optimal navigation and sensing operations of a swarm of drones $\mathcal{U}$ over all time periods $\mathcal{T}$ such that drones can efficiently and accurately collect required sensing data while minimizing their energy consumption.* The navigation and sensing of a drone u at time t is denoted by the plan $P^u(t)$. The efficiency and accuracy of sensing data collection as well as the energy consumption serve as the performance metrics.

The scenario of a swarm of drones that perform sensing is modeled, which considers the following performance metrics: (1) mission efficiency; (2) sensing accuracy; and (3) energy cost.

Mission efficiency. It denotes the ratio of sensing values in all cells at all timeslots collected by the drones during their mission over the total required values in all cells at all timeslots during the period t. The purpose is to collect sensing data as much as possible. It is formulated as:

$$\mathrm{Eff}(t) = \frac{\sum_{u=1}^{U}\sum_{n=1}^{N} V_n^u(t)}{\sum_{n=1}^{N} R_n(t)}. \tag{4.3}$$

However, maximizing this metric leads to sensing imbalance and even blind areas. Some cells may be covered for a long time while some other cells may be never covered, i.e., the over-sensing and under-sensing [22]. Over-sensing causes excessive data that needs further processing, waste of energy consumption, high storage and privacy cost, while under-sensing fails to satisfy sensing requirements. As a consequence, the generation and selection of high-quality plans are required to eliminate over-sensing and under-sensing. Thus, it is necessary to consider matching indicators such as the sensing accuracy [22] to access such imbalances.

Sensing accuracy. It denotes the matching (correlation[1]) between the total sensing values collected and the required ones. The metric is formulated as follows:

$$\mathrm{Acc}(t) = \sqrt{\frac{N \cdot S}{\sum_{n=1}^{N}[\sum_{u=1}^{U} V_n^u(t) - R_n(t)]^2}}. \tag{4.4}$$

Apart from improving the efficiency and accuracy of drone sensing, the energy consumed by drones needs to be saved.

Energy cost. It is the battery usage of drones to perform urban sensing. A power consumption model [93] is used to calculate the power consumption with input the specification of drones (weight, propeller and power efficiency). This model estimates the cost of navigation and sensing plans, and emulates the outdoor environments [26]. The energy cost of each drone u is formulated as:

$$E^u(t) = P^{\mathrm{f}}(u) \cdot t^{\mathrm{f}} + P^{\mathrm{h}}(u) \cdot \frac{\sum_{n=1}^{N} V_n^u}{f}, \tag{4.5}$$

where $P^{\mathrm{f}}(u)$ and $P^{\mathrm{h}}(u)$ denote the flying and hovering power consumption of u respectively; t^{f} is the total flying time of u without hovering, which is determined by the sensing plan $P^u(t)$; f is the frequency with which drones collect sensor data as they hover over a cell.

The goal of the proposed system is to find the optimal plan for each drone $u \in \mathcal{U}$, i.e., determining which cells a drone visits and how many sensor values it collects over a time period. The problem is formulated as follows:

$$\max_{V_n^u, u \in \mathcal{U}, n \in \mathcal{N}} \sum_{t=1}^{T} \mathrm{Eff}(t), \tag{4.6}$$

$$\max_{V_n^u, u \in \mathcal{U}, n \in \mathcal{N}} \sum_{t=1}^{T} \mathrm{Acc}(t), \tag{4.7}$$

$$s.t. \quad \sum_{u \in \mathcal{U}} \sum_{n \in \mathcal{N}} V_n^u \leq \sum_{n \in \mathcal{N}} R_n, \tag{4.8}$$

$$E^u(t) \leq C^u, \forall t \in \mathcal{T}. \tag{4.9}$$

Objective (4.6) measures the accomplishment of sensing tasks. Objective (4.7) meas-

[1]Error and correlation metrics (e.g. root mean squared error, cross-correlation or residuals of summed squares) estimate the matching, shown to be NP-hard multi-objective combinatorial optimization problem in this context [18, 39].

Algorithm 4: The local sensing plan generation strategy for each drone.

```
Input: Power consumption of drone $u$ for flying $P_u^{\mathrm{f}}$ and hovering $P_u^{\mathrm{h}}$, the battery capacity
       $C^u$, the targets $\boldsymbol{T} = (T_1, ..., T_N)$ and their coordinates, the base station $m$ for
       departure/return and its travel range $K^u$, the total number of visited cells $|J_u|$, the
       number of generated plans $K$.
1  Initialization: Initialize a set of plans $\widehat{\mathbb{P}} = \emptyset$
2  for each plan index $k := 1, ..., K$ do
3      Path calculation: Find $J_u$ via the K-nearest search within the range $K^u$
4      Find the shortest path among visited cells and base station $m$ via the greedy algorithm
5      Energy calculation: Calculate the flying energy $E^{\mathrm{f}}(J_u)$ based upon the path;
         Determine the energy utilization ratio $e$ based on $p$ via Eq.(4.15)
6      Calculate the maximum hovering energy $E^{\mathrm{h}}(J_u)$ via Eq.(4.16)
7      Calculate the total sensing values $V(J_u)$ to collect via Eq.(4.17) and the total targets
         $T(J_u)$ via Eq.(4.19)
8      Sensing allocation: Allocate the sensing values $V_n^u$ proportionally to the visited cells
         via Eq.(4.18)
9      Plan generation: Calculate the cost of the plan $E(J_u)$ via Eq.(4.20)
10     Generate the plan $\mathbb{P}$ of size $N \times M$, and add it to the set $\widehat{\mathbb{P}}$ of sensing plans with $E(J_u)$
11 end
   Output: Set of plans $\widehat{\mathbb{P}}$ for drone $u$.
```

ures the over-sensing and under-sensing. Equation (4.8) limits the total sensor value collected by drones to the required one at maximum. Constraint (4.9) models the energy constraint of the drone u, where $C(u)$ is the battery capacity of drone u.

4.3 Sensing Plan Generation Strategy

OPS optimizes sensing quality via a coordinated selection among alternative plans generated locally by the drones (i.e., navigation and sensing options). However, plans generation also results in a series of new problems, including *how to select the cells*, *how to determine collected sensing values*, *how to calculate the energy cost of a plan*, and *how to prevent more than two drones to occupy the same cell at the same time*. Therefore, a novel plan generation strategy is proposed to solve these problems such that drones coordinate efficiently to achieve high-quality sensing. Algorithm 4 outlines the following steps of the proposed plan generation strategy for a swarm of drones.

Initialization. Given a drone u, the inputs of the algorithm are listed: The flying and hovering power consumption are calculated based on the power consumption model [34].

The parameters of drone u $\{m_b, m_e, d, r, v, F_d, e\}$ and environmental parameters such as air density and air speed are determined (see Table 4.2 and Appendix A). Each drone u also needs the information of the map including the coordinates and sensing requirements of cells, as well as the base stations of departure and return.

Next, according to the objective functions of Eq.(4.7) and (4.6), the total number of visited cells $|J_u|$ is determined empirically using one of the three policies: (1) *max sensing accuracy*, (2) *max mission efficiency*, and (3) *balance*. Max sensing accuracy focuses on avoiding over-sensing and under-sensing, while max mission efficiency minimizes the uncollected sensing data. The policy for balance is a trade off between the first two policies. The policy of max sensing accuracy has a larger number of visited cells, while max mission efficiency has a lower one. The two theorems in Appendix B provide the theoretical foundations behind the design of these policies. The algorithm also initializes the set $\widehat{\mathbb{P}}$ for the plans of drone u.

Path calculation. At the beginning of each round, the algorithm finds the set of visited cells indices J_u via the K-nearest search. It selects the first cell randomly from a range of cells K^u, which indicates the flight range of drone u. This range is calculated based on the relative distance between base stations. Then, the algorithm searches the nearest cell (within K^u) to the previous selected one until finding other $|J_u| - 1$ cells. After this, the algorithm finds the shortest possible path among the cells of J_u and the base station m via the greedy algorithm for traveling salesmen problem [155], and calculates the traveling time τ_u excluding hovering. Note that the drone returns to m at the end of period, and thus the path starts and ends at m. Taking an example in Fig. 4.1, the first plan of Drone 1 has the set of visited cells $\{A, C, D\}$.

Power consumption model. Drones spend energy to surpass gravity force and counter drag forces due to wind and forward motions [93]. A drone controls the speed of each rotor to achieve the thrust T and pitch θ necessary to stay aloft and travel forward at the desired velocity while balancing the weight and drag forces. For a drone with mass m_b and its battery with mass m_c, the total required thrust is defined as follows:

$$\mathcal{T} = (m_b + m_c) \cdot g + F_d, \tag{4.10}$$

where g is the gravitational constant, and F_d is the drag force that depends on air speed and air density. For steady flying, the drag force can be calculated by the pitch angle θ as:

$$F_d = (m_b + m_c) \cdot g \cdot tan(\theta). \tag{4.11}$$

Building on the model in [34], the power consumption with forward velocity and forward pitch is given by:

$$P^{\mathrm{f}} = (v \cdot sin\theta + v_i) \cdot \frac{\mathcal{T}}{\epsilon}, \tag{4.12}$$

where v is the average ground speed; ϵ is the overall power efficiency of the drone; v_i is the induced velocity required for given T and can be found by solving the nonlinear equation:

$$v_i = \frac{2 \cdot \mathcal{T}}{\pi \cdot d^2 \cdot r \cdot \rho \cdot \sqrt{(v \cdot cos\theta)^2 + (v \cdot sin\theta + v_i)^2}}, \tag{4.13}$$

where d and r are the diameter and number of drone rotors; ρ is the density of air. Moreover, the power consumption for hovering of a drone is calculated by:

$$P^{\mathrm{h}} = \frac{\mathcal{T}^{3/2}}{\epsilon \cdot \sqrt{\frac{1}{2}\pi \cdot d^2 \cdot n \cdot \rho}}. \tag{4.14}$$

Energy calculation. The algorithm determines the maximum energy consumption E_u^{max} of the drone before calculating the collected sensor values and the corresponding energy cost. The algorithm uses the energy utilization ratio e to compute the maximum energy constraint $C^u \cdot e$. The ratio can be expressed as:

$$e = 1 - \frac{k}{\delta \cdot K}, \tag{4.15}$$

where δ is a constant to determine energy utilization. The ratio limits the energy consumption of drones, and gives them flexibility to select plans with varying cost, i.e. energy consumption. Next, the algorithm calculates the hovering energy consumption $E^{\mathrm{h}}(J_u)$ using the flying power consumption P_u^{f}:

$$\begin{aligned} E^{\mathrm{h}}(J_u) &= C^u \cdot e - E^{\mathrm{f}}(J_u), \\ &= C^u \cdot e - P_u^{\mathrm{f}} \cdot \tau(J_u). \end{aligned} \tag{4.16}$$

The total sensor values to collect among J_u visited cells $V(J_u)$ is then calculated as follows:

$$V(J_u) = \sum_{n \in \mathcal{N}} V_n^u = \frac{E^{\mathrm{h}}(J_u)}{P_u^{\mathrm{h}}} \cdot f, \tag{4.17}$$

where P_u^{h} is the power consumption for hovering. Both P_u^{f} and P_u^{h} are computed by the power consumption model [34] (see Appendix A).

Sensing allocation. To determine V_n^u in each cell n, the algorithm allocates the collected sensor values among J_u visited cells proportionally to the target; a higher number of sensor values is collected from the cell with a higher target value. The

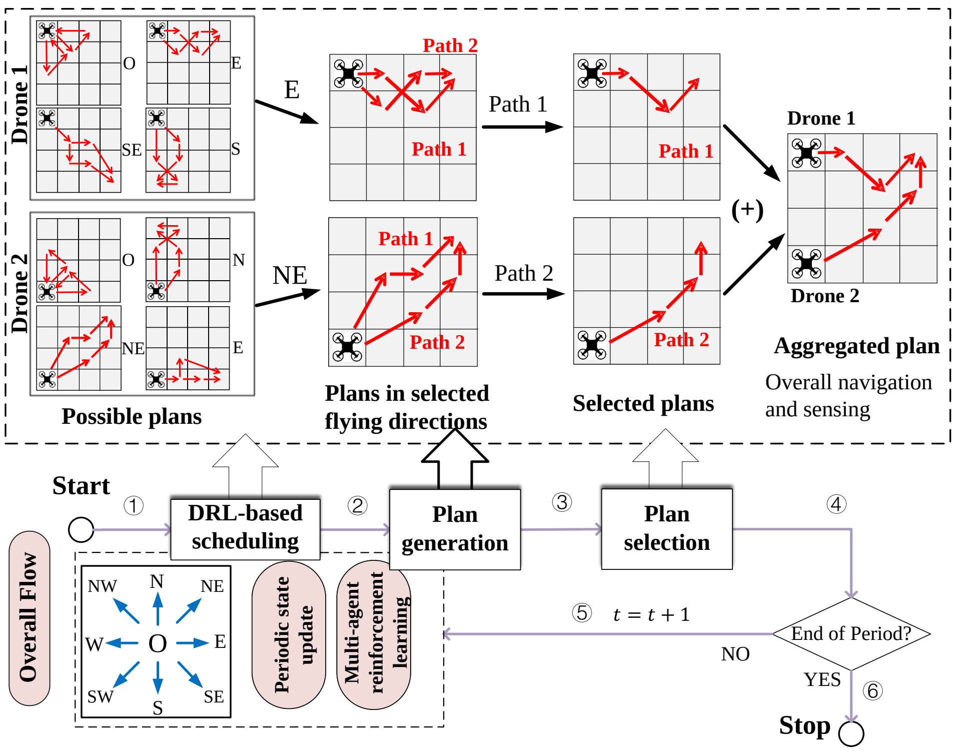


Figure 4.2: System framework of *HALOP* used in coordination of multi-drone navigation and sensing.

equation is shown as follows:

$$V_n^u = \begin{cases} V(J_u) \cdot \frac{R_n}{T(J_u)}, & n \in J_u \\ 0, & otherwise \end{cases}, \tag{4.18}$$

$$where \quad T(J_u) = \sum_{n \in J_u} R_n. \tag{4.19}$$

The proportional sensing allocation aims to improve the matching between the total sensor values collected and the required ones. Its high performance is also proved by comparing it to the equal allocation (mean) $V_n^u = \frac{V(J_u)}{|J_u|}$.

Plan generation. With the energy consumption of hovering and flying, the algorithm computes the cost of the plan $E(J_u)$:

$$E(J_u) = C^u \cdot e. \tag{4.20}$$

Finally, the algorithm generates the plan and adds it with the corresponding energy cost to the set $\widehat{\mathbb{P}}$.

4.4 Learning-based Approach for Sensing

This section aims to leverage the proposed approach of *HALOP* to the EAC-MDNS model. In this scenario, *HALOP* integrates distributed multi-drone coordination for short-term navigation and sensing optimization with long-term scheduling of flying directions. Fig. 4.2 illustrates the designed system framework of *HALOP* used in coordination of multi-drone navigation and sensing, consisting of three main components:

MARL-based scheduling. This component is the core of the framework, which leverages the MARL algorithm to enable drones to take the actions of overall flying directions between departure and destination charging stations in each time period. Thus, these actions are executed period-by-period, with each made only after the drone completes its sensing missions in the current period. The component includes two elements: a periodic state update, which updates the state of drones for MARL-based scheduling in the next period, and a multi-agent reinforcement learning module, which is built based on centralized training and decentralized execution.

Plan generation. This component generates the navigation and sensing options for drones under the chosen flying direction from the MARL-based scheduling component. Given the sensing map, each drone autonomously generates a finite number of discrete plans to initialize the overall process. This provides flexibility for the drones at the next stage to choose in a coordinated way.

Plan selection. This distributed component leverages multi-agent collective learning to coordinate drones to locally select the optimal navigation and sensing options from their generated plans within a period.

The *HALOP* process works as follows: Each drone initially generates a set of plans. It then selects a flying direction and chooses a subset of plans aligned with that direction. Each drone autonomously picks a plan from this subset, shares it with the swarm, and observes others' choices. Based on its action and observations, the drone calculates a reward function and updates its state (including location, battery level, and sensing requirements), and stores these results in a multi-agent reinforcement learning buffer to refine its flight strategies. This cycle of MARL-based scheduling and plan selection repeats at each time period until the mission is complete. However, *HALOP* requires frequent information sharing, which makes communication inefficient and vulnerable to potential failure of individual drones. The training process of *HALOP* applied in the

Algorithm 5: The *HALOP* training for multi-drone navigation and sensing.

```
1  Randomly initialize critic network $Q(\cdot)$, actor network $\pi(\cdot)$ with parameters $\theta^Q$, $\theta^\pi$
2  for episode := 1 to max-episode-number do
3      Reset the sensing requirements and the state of drones
4      for period $t := 1$ to max-episode-length do
5          for $\forall u \in \mathcal{U}$ do
6              Take action: $A_t^u = \pi(S_t^u|\theta^\pi)$
7              Find the flight range $K^u$ under the flying direction $A_t^u$
8              Generate plans through Algorithm 4
9              Select an optimized plan through collective learning [18]
10             Obtain the next state and reward via Eq.(4.21)
11             Store transition sample $(S_t^u, A_t^u, R_t^u, S_{t+1}^u)$ into buffer
12             Sample a random mini-batch of $H$ samples from buffer
13         end
14     end
15     Estimate advantage via Eq.(3.18)
16     Calculate the probability ratio via Eq.(3.19)
17     Update $\theta^\pi$ by minimizing the loss via Eq.(3.20) (3.21)
18     Update $\theta^Q$ by minimizing the loss via Eq.(3.22)
19 end
```

multi-drone navigation and sensing is illustrated in Algorithm 5. Here, the multi-agent reinforcement learning used in *HALOP* is illustrated in Section 3.4.3.

4.4.1 MARL modeling

The core design of *HALOP* lies in applying MARL on the multi-drone sensing problem. Once drones take actions of flying directions, they execute plans and return to charging stations, changing their current locations and battery levels while observing the navigation and sensing of other drones. Therefore, the problem scenario can be explained into a Markov decision process [156]. The problem is modeled using state, action, and reward concepts:

(1) *State*: The state $\mathbf{s}_t$ at period t consists of four components $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$, where $\mathcal{S}_1$ is the current locations of drones. Since drones charge at charging stations before taking actions, the location can serve as the index of the charging station. $\mathcal{S}_2$ is the current battery levels of drones, which are calculated based on the energy cost. $\mathcal{S}_3 = \{P^u(t)|_{u\in\mathcal{U}}\}$ is the selected plan of u. $\mathcal{S}_4 = \{P^{-u}(t)|_{u\in\mathcal{U}}\}$ is the aggregated plan of

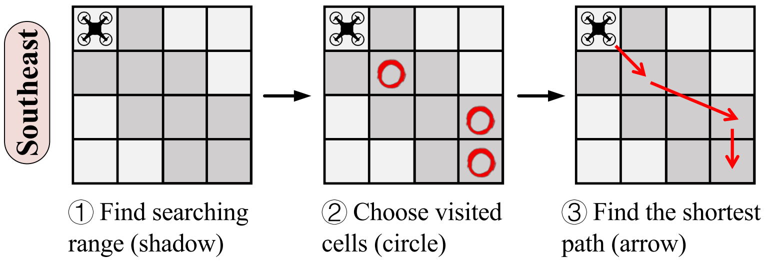

Southeast
① Find searching range (shadow)
② Choose visited cells (circle)
③ Find the shortest path (arrow)

Figure 4.3: Process of path finding in plan generation.

other drones excluding u, which are shared via the structured communication model.

(2) *Action*: To explain the long-term navigation and sensing over all periods, the period-by-period actions of each drone $\mathbf{a}_t = \{a_t^1, ..., a_t^U\}$ at period t are introduced, where $a^u = \{0, 1, 2, ..., 8\}$, $u \in \mathfrak{U}$. It means to control u to move horizontally along eight directions, which are 1 = north (N), 2 = east (E), 3 = south (S), 4 = west (W), 5 = northeast (NE), 6 = southeast (SE), 7 = southwest (SW), and 8 = northwest (NW), or return to the origin ($a^u = 0$). Under each action, u executes a short-term navigation and sensing. After completing its sensing tasks, u flies back to one of the charging stations to recharge fully and resume work in the next period $t + 1$.

(3) *Reward*: Based on the objective functions of Eq.(4.6) and (4.7), the expected immediate local reward of one drone at period t is defined as follows:

$$\mathbf{r}_t^u = \sigma_1 \mathrm{Eff}(t) + \sigma_2 \cdot \mathrm{Acc}(t) - \sigma_3 E^u(t), \tag{4.21}$$

where σ_1, σ_2 and σ_3 denote the tradeoff parameters to normalize and balance the mission efficiency, sensing accuracy and energy cost. Throughout the training process or episodes, the overall reward fluctuates based on the actions taken by the drones. This helps drones in prioritizing areas rich in sensor data by maximizing their reward, i.e., the highest cumulative overall performance.

4.4.2 Plan generation and selection

The action enable each drone to generate the plans under the corresponding flying direction. The plans generated by drone u during a period is defined as $G^u(t) \in \widehat{\mathbb{P}}$, where each plan has the same action, $P^u(t) \in G^u(t)$. Similar to the plan generation strategy in Section 4.3, a single plan is generated based on searching (flight) range K^u of drone u, number of visited cells within K^u, and the energy $E^u(t)$ consumed by u traveling over the visited cells, which is defined as $P^u(a^u, t) = \{k, K^u, J^u, E^u(t)\}$.

The searching range K^u is determined by a certain width (equal to half of distance to nearest charging station) along its flying direction, and then searches the cells within this range, see Fig. 4.3.

Given the plans in selected flying directions within a period, drones can adapt their task allocation by selecting appropriate plans in response to changes in task targets, enhancing their adaptability in dynamic environments. In this process, the agents of drones improve their plan selections via tree communication. Specifically, each agent obtains the aggregated choices, i.e., the aggregated plan $P^{-u}(a^u, t)$, from other agents, and chooses one of the plans $P^u(a^u, t, l)$ such that all choices together $P^u(a^u, t) + P^{-u}(a^u, t)$ add up to match a target $\tau(t)$. The target is used to steer each drone to sense over a unique cell during a timeslot, thereby avoiding the simultaneous sensing of multiple drones within the same cell, i.e., preventing over-sensing and under-sensing.

After generating plans, each drone coordinates to select its optimized plan, indicating a specific path within a period (comprising multiple timeslots). The plan selection utilizes multi-agent collective learning in the low-level layer. More details will be illustrated in the algorithm settings of Section 4.5.

4.4.3 Periodic state update

After choosing a plan, each drone changes its current state at the next period $t + 1$, including its location, battery level, selected plan $P^u(t+1) := P^u(a^u, t)$, and observed plan $P^{-u}(t+1) := P^{-u}(a^u, t)$. However, due to dynamic changing environment, drones have no knowledge about the required sensing value in the next period. They need to predict and estimate the required sensing value to calculate reward function via Eq.(4.3) and (4.4). Depending on the predicted distribution of sensor data, the target is required to be updated.

Predicted sensing value. The predicted sensing value at period t, denoted as $\hat{V}(t)$, is updated in a time-reverse decay, formulated as follows:

$$\hat{V}_{ns}(t) = \sum_{t'=1}^{t} (T - t + t') \cdot \omega_{ns}(t') \cdot v'_{ns}(a^u, t') := V_{ns}(t), \tag{4.22}$$

where V'_{ns} denotes the data values collected by drones once they execute their selected plans; $\omega_{ns}(t')$ is a prediction coefficient such that $0 < \omega_{ns}(t') < 1$, $t' \leq t$. To achieve accurate predictions, *HALOP* leverages the Ordinary Least Squares regression method

(OLS) to train these coefficients initially, and use the past experienced observations as the target distribution for training.

Target. The target τ in the plan selection needs to be iteratively updated to steer the coordination of drones to choose plans for areas and timeslots with abundant sensor data. The percentile-based data filtering is used to eliminate the extreme low sensing values collected by drones. This approach effectively removes the need for data collection in regions with low sensing requirements (e.g., traffic exclusion zones). As a consequence, it helps drones in prioritizing their operations in cells and time where sensing values are significantly high. The target is initialized as $\tau_{ns}(0) = 1$ to encompass all cells and timeslots, and update it as follows:

$$\tau_{ns}(t) = \begin{cases} 0, & r_{ns}(a^u, t) < \overline{v} \wedge g_{ns}(t) > 0 \\ \tau_{ns}(t-1), & otherwise \end{cases}, \tag{4.23}$$

where the threshold $\overline{v}$ is set iteratively and calculated as the value at the $100(1-U/N)$th percentile among the predicted data values $\hat{V}_{ns}(t)$.

At the end of state update in *HALOP*, the current collected data values V'_{ns}, the predicted values $\hat{V}_{ns}$ and the target τ are shared with each agent via the top-down interactions within the tree communication structure. This information will be stored and later sampled for multi-agent reinforcement learning.

4.5 Experimental Evaluation

This section introduces the experimental settings including the sensing map, the drones and the algorithms at first. The metrics and baselines used for the performance evaluation are also introduced.

4.5.1 Experimental settings

Static sensing scenarios. This scenario assumes a static sensing environment over a long time span in order to evaluate the *OPS* approach. In other words, the repeated use of individual drones is not considered-only the total number of drones dispatched is taken into account. Each drone can be deployed at any time during the day, and there is no requirement to monitor temporal changes in the sensing environment. In this scenario, a square area of size 1600×1600 meters split into a finite number of cells is studied. Each cell is defined as a rectangular square that can be captured by the cameras of drones (see Fig. 4.1). The target values, or the sensing requirements of hovering time are distributed according to a Beta distribution. The base (charging) stations are uniformly distributed in the map. There are three types of scenarios for the experimental evaluation:

- *Basic static sensing scenario.* It has 4 base stations, 64 cells and the total target values of 20000 to collect over 48 time periods, which correspond to one day. Each period lasts 30 minutes and is divided into 12 time units, each of equal length. This scenario dispatches 1000 drones over all periods, during which approximate 20 drones sense the area (camera recording) in parallel. The purpose to use this map is to compare the performance of different plan generation policies in the proposed method.
- *Complex static sensing scenario.* It varies the parameter settings such as the number of dispatches, the number of base stations and cells as well as the total target values. The goal is to assess the scalability of the proposed method in different experimental conditions.
- *Static transportation scenario.* It originates from the central business district of Athens, where a swarm of drones use cameras to record traffic flows. The goal is to assess the accuracy and efficiency of the proposed method in the real-world traffic monitoring. More details are given in Section 4.5.4.

For the validation of the proposed algorithm, a number of 200 sensing scenarios (in basic and complex) are generated, each with a new distribution of cells and target values.

Evolving transportation scenarios. This scenario aims to simulate a realistic and evolving sensing environment. In order to evaluate the *HALOP* approach, a real-world transportation scenario is modeled, where a swarm of drones perform the sensing tasks of traffic monitoring. The number of vehicles serves as the required sensing value in the model. The experimental scenario is based on a detailed traffic flow of Munich, Germany. A realistic traffic distribution is generated using real-world traffic flow data[1] representing actual vehicle journeys within the city [153], integrated with the SUMO (Simulation of Urban Mobility) traffic simulator[2]. To model dynamic traffic behavior

[1]Munich IoT Benchmarking Dataset, available at: https://github.com/Znbne/MunichIoT.
[2]SUMO, available at: https://www.eclipse.org/sumo/.

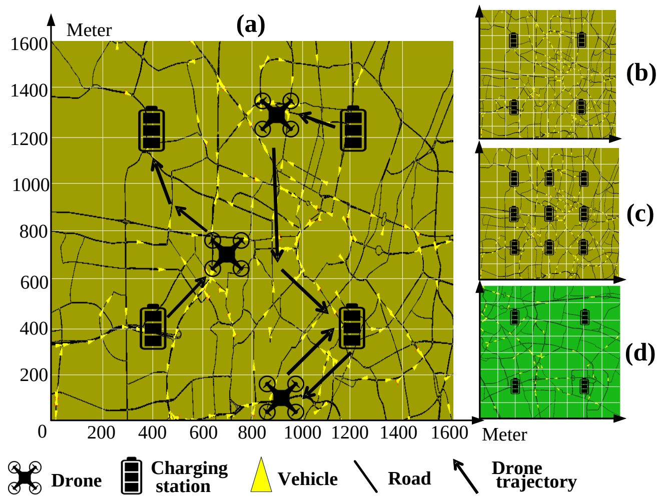


Figure 4.4: Multiple scenarios in central business district of Munich, Germany.

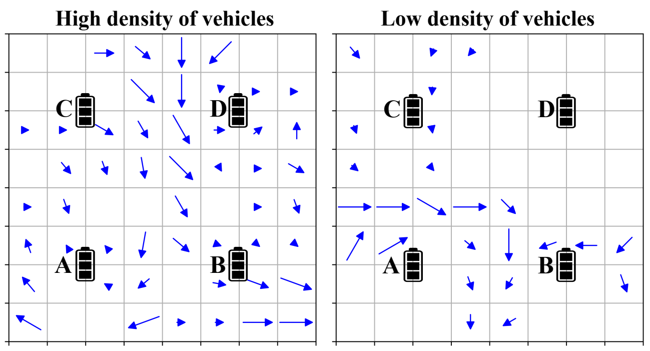


Figure 4.5: The distribution of both charging stations and traffic vehicles in the maps with high and low density of vehicles.

and improve the realism of vehicle routing, the duaIterate algorithm[1] is employed to iteratively refine the vehicle routes to optimize traffic flow and mitigate congestion [153, 154].

Fig. 4.4 illustrates a selected map of 1600×1600 meters in the city with the simulation time of 50 hours, which is 100 periods. It has a high density of vehicles, approximately $2,000$ vehicles passing by per hour. The area split into a finite number of cells, each is defined as a rectangular square with the size of 200×200 meters that can be captured by the cameras of drones. Fig. 4.5 shows the distribution of both char-

[1]Duarouter, available at: https://sumo.dlr.de/docs/duarouter.html.

Table 4.2: Notations for sensing drones.

Notation	**Value**
Mass of drone body	$1.07kg$
Mass of battery	$0.31kg$
Diameter of propellers	$0.35m$
Number of propellers	4
Ground speed	$6.94m/s$
Power efficiency	$e = 0.8$
Battery capacity	$C^u = 160kJ$
Sensing frequency	$f = 60$ *seconds*

ging stations and traffic vehicles in the map. There are 4 charging stations uniformly distributed in the map with $64 = 8 \times 8$ cells lined up over the map. The blue arrows symbolize the flow of vehicular traffic, with their length proportional to the volume of vehicles over 8 periods. The charging stations are uniformly distributed in the map. The cross-validation is employed: 80% simulation time of the datasets for training and 20% for testing. There are two scenarios for the experimental evaluation:

- *Basic evolving transportation scenario.* It has 64 cells, 4 charging stations and 8 time periods (set as $30min$ for each period). It has high density of vehicles, around $2,000$ vehicles passing by per time period. 16 drones sense the area (camera recording) in parallel.

- *Complex evolving transportation scenario.* It varies the parameter settings such as the density of drones and vehicles, as well as the number of time periods, cells and charging stations. Therefore, multiple scenarios are defined as shown in Fig. 4.4: (a) Basic evolving transportation scenario with 64 cells, 4 charging stations and high density of vehicles; (b) Increase the number of cells to 100; (c) Increase the number of charging stations to 9; (d) Change to a new map with low density of vehicles.

Drones. The drones used in the scenarios are of the same type (DJI Phantom 4 Pro), equipped with the same type of battery (6000 mAh LiPo 2S) and camera (4K) to capture images/videos[1], and thus they have the same power consumption [34] and

[1]https://www.dji.com/uk/phantom-4-pro/infospecs

battery capacity. To ensure the camera of a drone covers the whole area of a cell (see Fig. 4.1), the minimum hovering height of drones is determined at which the field of view of the camera and the cell overlap. Based on the distance between any two cells D (approximately 200 meter in the basic static sensing scenario), the hovering height H is computed using the pixels PX, focal length derived from the camera calibration CC and ground sampling distance GSD, with the formula: $\mathrm{H} = \mathrm{GSD} \cdot \mathrm{CC}/\mathrm{PX}$ [157]. Thus, each drone is equipped with a $4K$ camera, sensing from a minimum height of 164.8 meter based on the pixels and field of view that a $4K$ camera has. The drone parameters and their description are summarized in Table 4.2.

Performance metrics. To evaluate the sensing quality, i.e., the accomplishment of sensing targets, three performance metrics are introduced:

- *Energy cost.* It denotes the total energy consumption of drones that execute the sensing missions defined by their selected plans over all time periods. It is formulated as Eq.(4.5).
- *Sensing accuracy.* It denotes the matching (correlation) between the total sensing values collected and the target values. A high sensing accuracy prevents the cases of over-sensing and under-sensing [26]. It is formulated as Eq.(4.4).
- *Mission efficiency.* It denotes the ratio of sensing values in all cells that are collected by the drones during their mission over the total target values in all cells. It is formulated as Eq.(4.3).

In simple words, the sensing accuracy measures the data sampling quality, while the mission efficiency measures the completeness of the required collected data. Furthermore, to obtain a comprehensive assessment that considers all three metrics, an overall performance evaluation is conducted using Eq.(4.21).

4.5.2 Algorithm settings and baselines

To ensure the decentralization and scalability, both *OPS* and *HALOP* used in sensing scenarios leverage a multi-agent collective learning algorithm of I-EPOS [18, 39] in the plan selection. Its purpose is to match the aggregated plans of all agents to the target while minimizing the energy cost $E(a^u, t)$ of the plan selected by the drone. The cost

Table 4.3: Parameters of the I-EPOS algorithm.

Parameters	**Value**
Number of agents/drones	1000
Number of plans per agent	64
Network communication topology	balanced binary tree
Number of repetitions	40
Number of iterations	40
Non-linear cost function	Min RMSE
Energy utilization parameter	$\delta = 8$
Behavior of agent	$\beta^u = 0$
Number of tested maps	200

function for each agent is formulated using the root mean square error (RMSE):

$$\min_{a^u, u \in \mathcal{U}} (1 - \beta^u) \cdot \sqrt{\frac{\sum_{n=1}^{N} \sum_{s=1}^{S} [g_{ns}(t) - \tau_{ns}(t)]^2}{N \cdot S}} + \beta^u \cdot E(a^u, t), \tag{4.24}$$

where β^u represents the behavior of a drone. As the value of β^u increases, the drone becomes increasingly selfish, prioritizing plans with lower energy costs at the expense of higher root mean squared error. Each drone aims to minimize the system-wide cost of sensing while considering its energy consumption.

The details of algorithm settings in both collective and reinforcement learning are illustrated as follows:

Economic planning and optimized selection. During the coordinated plan selection via I-EPOS, agents, which are mapped to drones, self-organize into a balanced binary tree as a way of structuring their learning interactions [18]. The algorithm repeats 40 times by changing the random position of the agents on the tree[1]. At each repetition, the agents perform 40 bottom-up and top-down learning iterations during which RMSE converges to the minimum optimized value. Table 4.3 summarizes the optimization parameters of I-EPOS.

Neural network and learning algorithm. In the reward function, the tradeoff parameters σ_1, σ_2 and σ_3 are all set to a value of 1. The proximal policy optimization is used in *HALOP* to improve the stability of learning process [135]. A total of 64 groups of transitions are sampled as mini-batches in a replay buffer, with a discount factor of

[1]More information about the influence of the tree topology and agents' positioning on the tree is illustrated in earlier work [64, 158].

0.95 and a clip interval hyperparameter of 0.2 for policy updating. The algorithm uses the recurrent neural network (RNN) with $W = 64$ neurons in the two hidden layers of the RNN in both critic and actor networks. The activation function used for the networks is tanh. The models are trained over 5000 episodes, each consisting of multiple epochs (equal to the number of time periods).

A fair comparison of the proposed methods with related work is not straightforward as there is a very limited number of relevant decentralized and learning algorithms. These algorithms [29, 48] cannot be directly applied to this large-scale task allocation problem while respecting the energy constraints of drones. For this reason, four state-of-the-art centralized sensing methods capable of performing multi-drone task optimization are used to compare with the proposed *OPS*: *Greedy-sensing* [159], *Round-robin* [160], *Min-energy* [18], and *MAPPO* [137]:

- *Greedy-sensing.* It requires a drone to complete the required sensing tasks of the cells one by one without violating the battery constraint. This method reduces the number of visited cells and traveling distance compared to the proposed method; drones spend more energy on sensing than traveling. The method is implemented on a centralized coordinator has a global view of the remaining sensor values required such that over-sensing and under-sensing are prevented. Table 4.4 illustrates the higher performance of the method with a global view vs. a version with a local view, i.e., no knowledge of the remaining sensor values required.

- *Round-robin.* It comes in sharp contrast to *Greedy-sensing.* Drones visit the same number of cells and spend more energy on traveling than *Greedy-sensing.* According to the results shown in Table 4.5, the number of visited cells is divided equally into 8 for each drone as it has the minimum sensing accuracy.

- *Min-energy.* It minimizes the total energy consumption and does not sacrifice energy for improving sensing quality. This method is implemented by changing the behavior of agents to $\beta = 1$ such that the agents select the plans with the lowest energy consumption cost. No coordination is performed in this case.

- *MAPPO.* It is a state-of-the-art MARL algorithm using PPO [99, 137], but does not include distributed optimization compared to *HALOP*. Moreover, agents in *MAPPO* learn the flying directions timeslot by timeslot, rather than period by period as in *HALOP*. At each timeslot, a drone takes actions to horizontally move

Table 4.4: Performance of two implementations in *Greed-sensing*.

Implementation	**Global view**	**Local view**
Sensing Accuracy	0.36	0.31
Mission Efficiency (%)	80.20	73.89

Table 4.5: Results for the number of visited cells in *Round-robin*.

Number of Visited cells:	**5**	**6**	**7**	**8**	**9**	**10**
Sensing Accuracy	0.45	0.42	0.43	0.54	0.33	0.26
Mission Efficiency (%)	74.07	59.58	37.41	24.49	14.31	3.32

to an adjacent cell in eight directions or hover (similar to the actions in $HALOP$), and returns to the nearest charging station at the end of every S timeslots (one period). For fair comparison, this method employs the same reward function and structured tree communication model to share the aggregated observation.

4.5.3 Evaluation on static sensing scenarios

The results of basic static sensing scenario using *OPS* is evaluated at first. As shown in Fig. 4.6, the three optimization methods based on *OPS* coordinate drones to select the plans with the minimum energy utilization ratio e and result in the lowest total energy consumption except *Min-energy* ($\beta = 1.0$) and *OPS-Pareto* ($\beta = 0.2$). The coordination of drones achieves the highest sensing accuracy compared to baselines. Therein, *OPS-balance* (with a total energy consumption of $157,944kJ$ and sensing accuracy of 0.49) has the mission efficiency of 80.4% that is close to *Greedy-sensing*; *OPS-accuracy* sacrifices mission efficiency to obtain a very high sensing accuracy among three policies, and just 0.02 lower than *Round-robin*; *OPS-efficiency* has the highest mission efficiency of 87.51% among all methods. In contrast, without coordination, the *Min-energy* method lowers energy consumption to $135,845kJ$ at a cost of lower sensing accuracy (0.27) and mission efficiency (52.5%). *Greedy-sensing* and *Round-robin* also come with lower sensing accuracy (0.36) and mission efficiency (14.49%) respectively. In overall, the proposed method is superior to baseline methods especially when combining all performance metrics.

In the complex static sensing scenario, there are four dimensions are studied here:

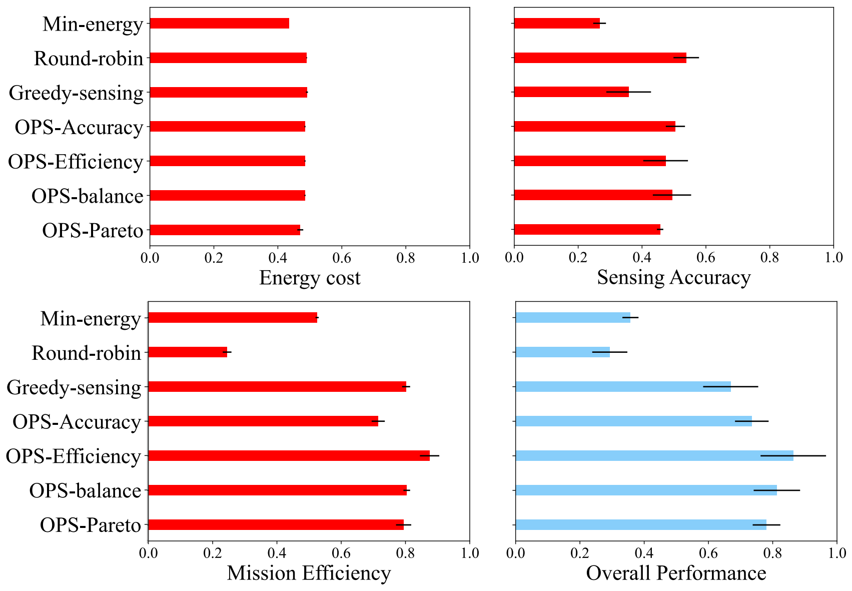


Figure 4.6: Performance comparison of the six methods on the basic static sensing scenario: 4 base stations, 64 cells and 20000 total target values.

(1) the number of dispatched drones used in the sensing mission (from 200 to 1000), (2) the total target values to collect (10000 and 20000), (3) the number of cells into which the same map is split (64 and 128), and (4) the number of base stations (heatmap triplets: up=4, down=16). Fig. 4.7 shows the performance comparison between *OPS-balance*, *Greedy-sensing* and *Round-robin* methods.

Number of dispatches. Take an example of the complex static sensing scenario (target=20000, cells=64, base stations=4). As the number of dispatches increases, the total energy consumption of *OPS-balance* rises, while mission efficiency decreases proportionally (from $157,944kJ$, 83% to $31,574$, 19% in Fig. 4.7). This shows that drones take full advantage of energy resources to collect sensor values efficiently. Compared to *Round-robin*, *OPS-balance* collects more data (below the average of 38.5%) with a lower number of dispatches and energy. Furthermore, as the number of dispatches decreases, *OPS-balance* shows a lower decrease of sensing accuracy (decreased by 4.21%) than *Greedy-sensing* (decreased by 26.17%). If some drones fail to be dispatched due to attacks or other factors, *OPS-balance* is proved to effectively mitigate the penalties of over-sensing and under-sensing, which validates the resilience of the proposed method.

Total target values. If the total target values to collect decrease from 20000 to

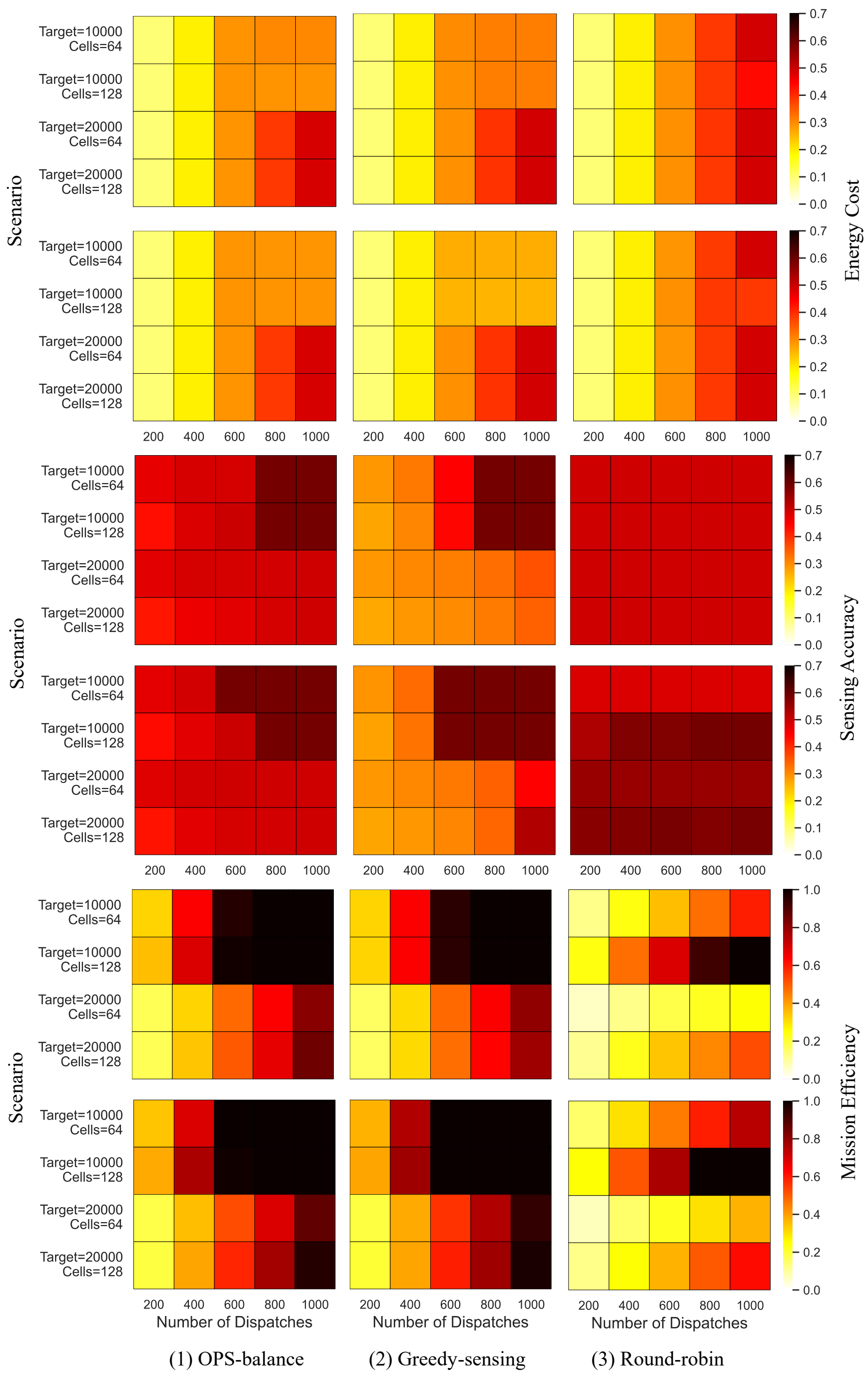


Figure 4.7: Performance comparison under varying parameters: total target values, the number of cells, the number of base stations, and the number of drone dispatches.

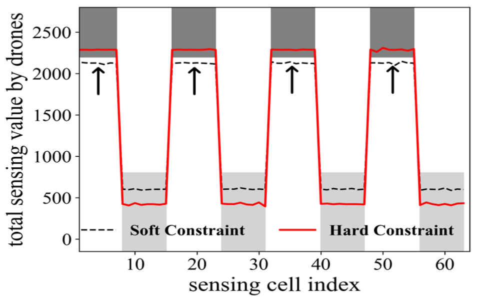
10000, the mission efficiency of *OPS-balance* reduces to 0 and the energy consumption keeps constant when the number of dispatches exceeds 500. This is because the total target values represent the total hovering time of drones, and *OPS-balance* does not waste energy resources when drones complete their sensing tasks. In Fig. 4.7, the sensing accuracy of *Greedy-sensing* is highly sensitive to the change of target values. It increases by 0.18, while *OPS-balance* increases only by 0.07.

Number of cells. If the number of cells increases from 64 to 128, the size of each cell decreases, and the distance between any two cells decreases, which reduces the travel distance of drones. As a consequence, drones spend more energy on collecting sensor values according to Eq.(4.16) and Eq.(4.17), which enhances the mission efficiency. In Fig. 4.7, *OPS-balance* shows a higher increase in mission efficiency (increased by 4%) than *Greedy-sensing* (increased by 1%) as the number of cells increases. Furthermore, *OPS-balance* collects higher quality data with a lower number of cells and drones as shown in Fig. 4.7.

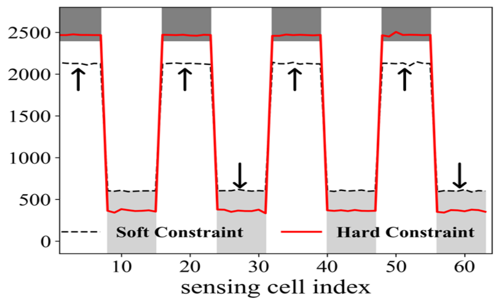
Number of base stations. If the number of base stations increases from 4 to 16, the travel distances of drones reduce and the total sensor values collected rise. As a result, accuracy and efficiency of *OPS-balance* increase. *OPS-balance* shows an overall superior performance as targets and base stations vary.

In summary, *OPS-balance* has the highest sensing accuracy (close to *Round-robin*) and mission efficiency (close to *Greedy-sensing*) as the variables in the environment change. This confirms that *OPS-balance* has the highest performance in overall under the same battery constraints and different sensing scenarios.

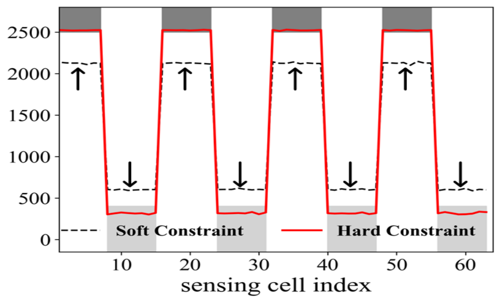
Hard constraint satisfaction of drone sensing. To test the hard constraint satisfaction of drone sensing scenario, three incremental levels of hard constraints are set to the aggregate choices (total sensing). These levels are quantiles chosen empirically by observing the median global plan after several executions of I-EPOS based on soft constraints. The agents are assumed here altruistic, such that: $\beta^u = 0, \forall u \in \{1, ..., U\}$. In this scenario, an upper bound $\mathcal{U}_d$ of hard constraints may represent privacy-sensitive areas or regulated no-fly zones for drones. In contrast, a lower bound $\mathcal{L}_d$ may represent minimal information required to monitor effectively a phenomenon, e.g. a forest fire or traffic jam. Fig. 4.8 shows the aggregated plans for the soft constraint (baseline) along with three incremental and alternating levels of hard constraints (upper/lower bounds). Light-grey shaded areas represent the upper bound and dark-grey shaded



(a) Upper bound = 800, lower bound = 2200, satisfaction rate = 1, sensing accuracy = 0.1251.



(b) Upper bound = 600, lower bound = 2400, satisfaction rate = 0.58, sensing accuracy = 0.1241.



(c) Upper bound = 400, lower bound = 2500, satisfaction rate= 0.055, sensing accuracy = 0.1139.

Figure 4.8: Optimization under soft and three levels of hard constraints in the drone swarm sensing scenario.

areas the lower bound. Arrows point to violations of hard constraints. Under soft constraints, the upper and lower bounds are violated, whereas hard constraints prevent these violations. Stricter hard constraints prevent more violations, however, the satisfaction rate drops significantly, while the sensing accuracy decreases. This shows that strict hard constraints are likely to oppose the soft constraints.

4.5.4 Vehicle observation using static transportation data

To further evaluate the *OPS* on drone sensing, this study uses the real-world data of pNEUMA from vehicle trajectories collected by a swarm of drones in the congested downtown area of Athens, Greece [4]. This application scenario envisions a large-scale and long-term monitoring of traffic congestion using coordinated drones to measure

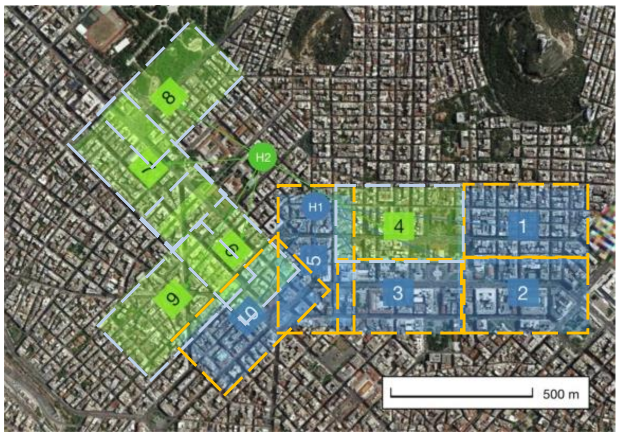


Figure 4.9: A swarm of 10 drones hovering over the central business district of Athens over five days to record traffic flows.

and predict traffic patterns. As shown in Fig. 4.9, a swarm of 10 drones hover over 10 cells for 20 time periods to record traffic streams of 6 types of vehicles: *car*, *taxi*, *bus*, *medium vehicle*, *heavy vehicle* and *motorcycle*. Each drone departs from and returns to one of the two base stations and hovers up to 25 minutes at a single cell.

The application scenario of coordinated drones that perform traffic sensing is modeled. To improve the sensing quality, the required sensing tasks (or targets) is calculated to be proportional to the spatio-temporal normalized distribution of vehicles shown in Fig. 4.10. The typical distributions can be derived from historical data[1]. The higher the likelihood a vehicle type drives through a cell, the higher the target value is, and $R_n \leq (25 \times 20)$, $n = 1, 2, ..., 10$. Over 20 time periods (30 min), a certain number of drones in each period is dispatched.

Fig. 4.10 illustrates the performance comparison between the proposed *OPS-balance* and the baseline for varying numbers of dispatched drones used in traffic monitoring. The results show that the accuracy of *OPS-balance* among six types of vehicles is significantly higher and more stable than that of *Greedy-sensing* when the number of dispatches is lower than 200. This is because with the total of 200 drones' dispatches under *Greedy-sensing*, each cell within each period is monitored by exactly one drone. With a lower number of drones' dispatches (i.e., scarce resources), however, *OPS-balance* coordinates drones to monitor all cells over all time periods, preventing over-sensing and

[1] https://open-traffic.epfl.ch/

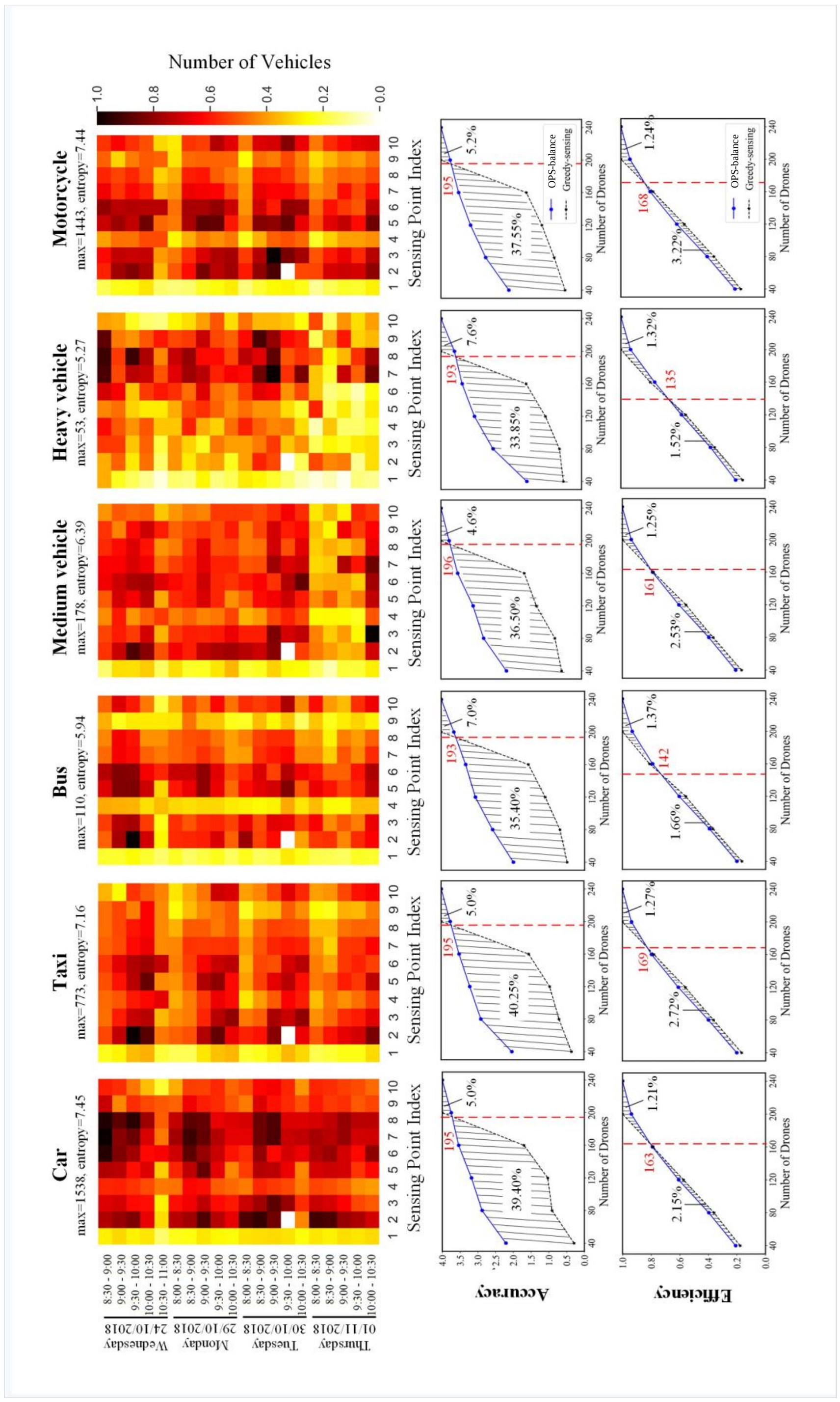


Figure 4.10: The comparison results among six types of vehicles in a downtown area of Athens based on the open traffic monitoring data collected by a drone swarm.

under-sensing. Fig. 4.10 also illustrates that *OPS-balance* is relatively more efficient than *Greedy-sensing* with no more than 160 dispatches. This is because *OPS-balance* coordinates drones to collect sensor values that are proportional to the distributions of vehicles, which increases the number of vehicles observed by drones. Note that there is a strong linear relationship between the threshold for the number of dispatches and the entropy of vehicles distribution, with a Pearson coefficient correlation of 0.93 and corresponding p-value of 0.007.

In summary, for lower than 160 **dispatches, *OPS-balance* is approximately** 46.45% **more accurate and** 2.88% **more efficient than *Greedy-sensing* among six vehicle types monitored.** This verifies the remarkable performance of the proposed method under a scarce number of drone resources, requiring less than 80% of the drones to achieve equivalent or higher performance than *Greedy-sensing*.

4.5.5 Evaluation on evolving transportation scenarios

This section aims to validate the effectiveness of the proposed approach *HALOP* on the designed multi-drone coordination model on urban sensing. To evaluate *HALOP*, the transportation sensing map based on traffic flows in Munich is used. Except for *MAPPO*, the proposed approach is compared with *OPS* that uses the strategy of *balance*.

There are six dimensions for the complex evolving transportation scenario are studied here: (1) the density of drones, (2) the number of periods, (3) the number of cells, (4) the number of charging stations, (5) the density of vehicles, and (6) the number of drones and cells while fixing the density of drones. Results are shown in Fig. 4.11-4.15, where the shadow area around the lines represents the standard deviation error of the results.

Density of drones. It denotes the ratio of the number of drones over the number of cells, representing the coverage of a single drone over the map. In this case, the number of cells is fixed as 64, and increase the number of drones from 8 to 64, that is, the drones density increases from 0.125 to 1.0. Fig. 4.11 illustrates the performance of *HALOP* and baseline methods when using different drones density to perform traffic monitoring. As drones density increases, both mission efficiency and sensing accuracy of *HALOP* increase linearly. These are approximately 23.0% and 15.8% higher than

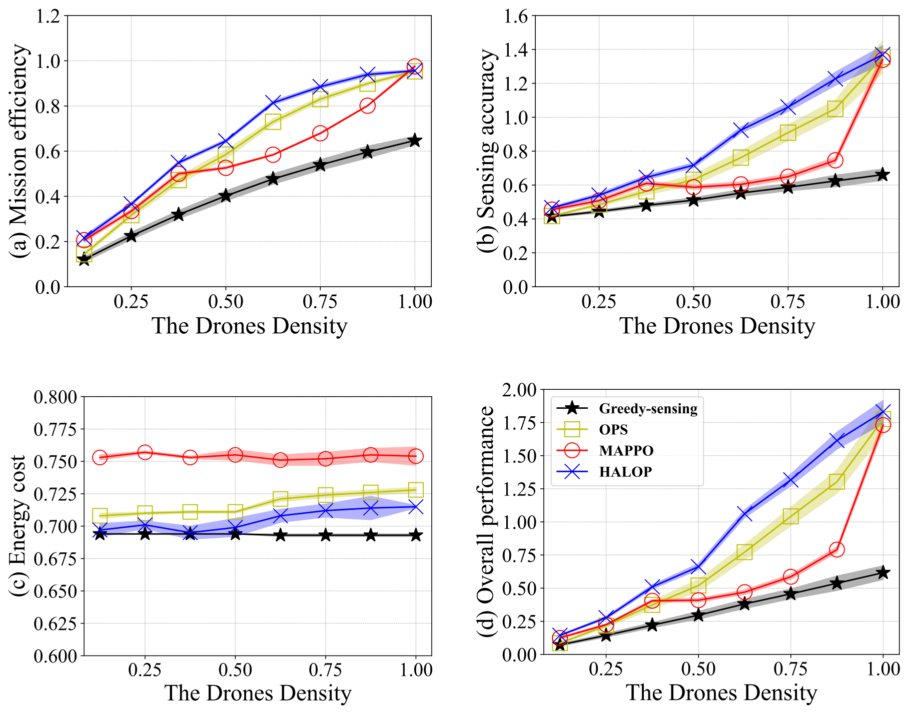


Figure 4.11: Changing the drones density by increasing the number of drones from 8 to 64 and fixing 8 periods, 64 cells, 4 charging stations and high density of vehicles.

OPS respectively, as shown in Fig. 4.11(a) and Fig. 4.11(b) [1]. This is because *HALOP* controls the traveling direction of drones based on the predicted sensor data, so that drones cover the area with the maximum sensing value. *MAPPO* has high efficiency and accuracy with a low density of drones (maximum p-value less than 0.001 using Mann-Whitney U test), but its performance degrades when the drones density is higher than 0.375 due to the high computational complexity. In contrast, the plan selection in *HALOP* reduces the action space and mitigates the learning difficulty, resulting in high performance with a high number of drones. In Fig. 4.11(c), the energy cost of *HALOP* is on average 1.7% lower than *OPS* and 7.9% lower than *MAPPO*. *Greedy-sensing* has the minimum energy cost among all methods, only 7.89% lower than *HALOP*, but has the lowest performance in efficiency and accuracy. *HALOP* sacrifices a little energy

[1] *HALOP*, *OPS* and *MAPPO* have similar overall performance when drones density is 1.0, with maximum p-value of 0.04, because there are ample drone resources available to effectively coordinate the coverage of the map.

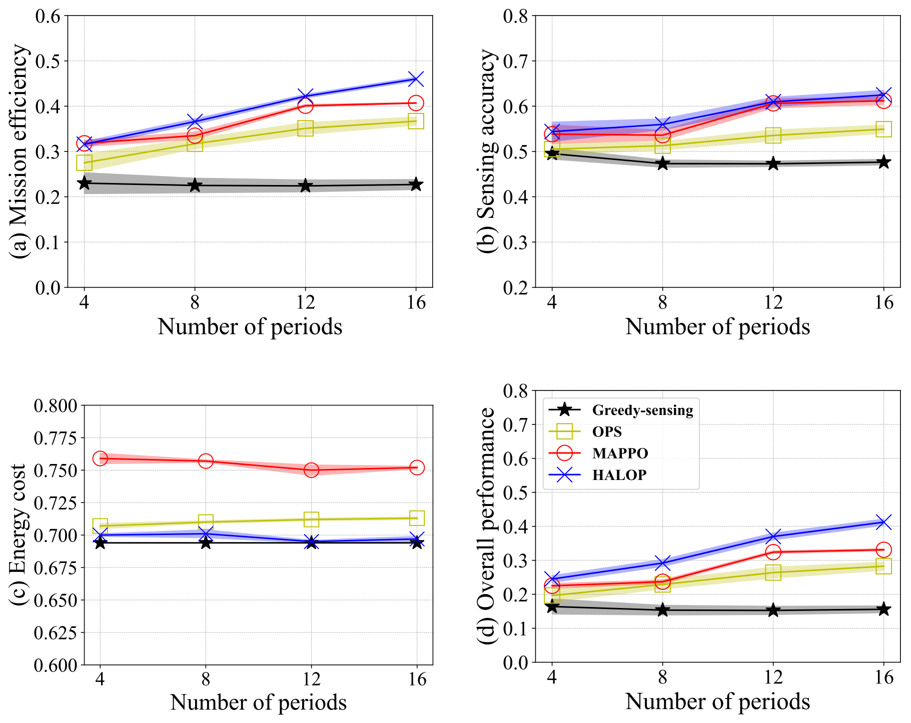


Figure 4.12: Changing the the number of periods from 4 to 16 and fixing 16 drones, 64 cells, 4 charging stations and high density of vehicles.

for sensing, but still gets lower cost than other methods. The results show that *high density of drones increases both mission efficiency and sensing accuracy of all methods, especially for short-term optimization methods (*HALOP *and* OPS*).*

Number of periods. As shown in Fig. 4.12, *HALOP* achieves superior performance compared to other methods, increasing linearly as the number of periods increase (increased by 51.3% in quadruple time). Although it has statistically similar mission efficiency to *MAPPO* when the number of periods is 4, this metric increases dramatically because drones update and obtain more accurate predicted sensor data with higher number of periods. The timeslot-by-timeslot learning in *MAPPO* perplexes the actions learned by drones compared to *HALOP*, since drones only take actions once per period (30min) in *HALOP* but take actions per minute in *MAPPO*. This also results in higher flying energy consumed by drones, and thus the energy cost of *MAPPO* is around 8.1% higher than *HALOP*. The results show that *high number of periods increases both mission efficiency and sensing accuracy of all methods, especially for long-term learning*

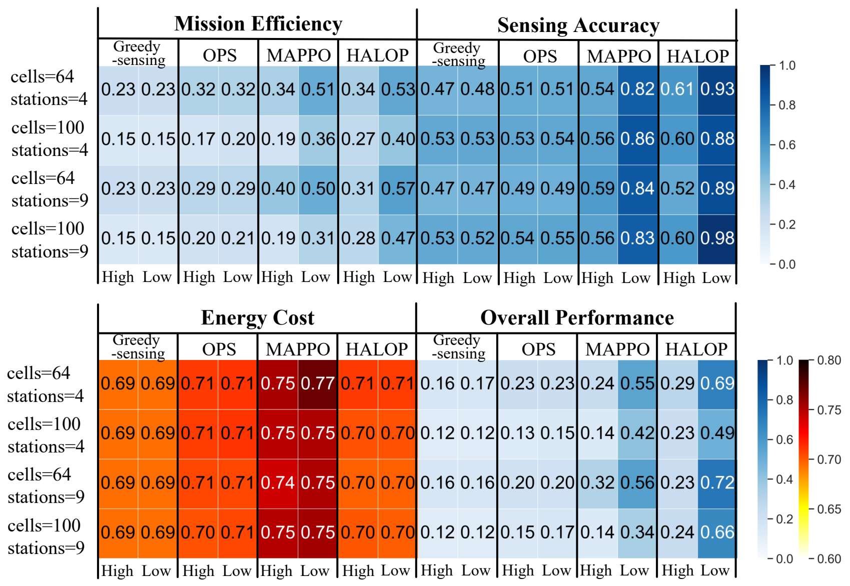


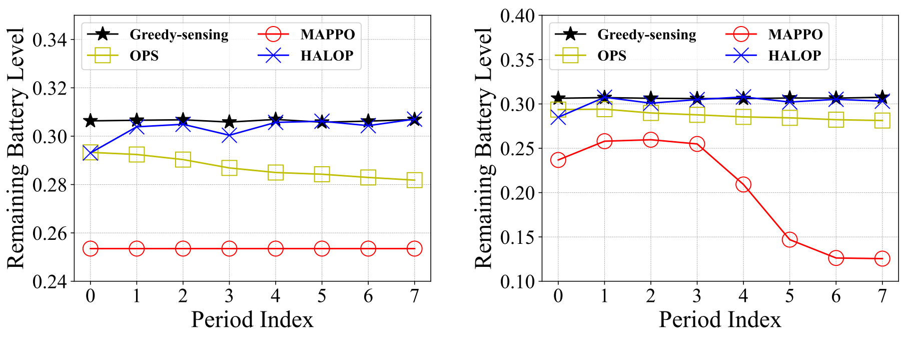
Figure 4.13: Performance comparison under varying parameters: the number of cells (64 and 100), the number of charging stations (4 and 9), and the density of vehicles (high and low).

*methods (*HALOP *and* MAPPO*).*

Fig. 4.13 illustrates the performance comparison between *HALOP* and baseline methods, varying the number of cells, charging stations and the density of vehicles, while keeping the number of drones fixed at 16 and periods at 8. The red palette on the right represents the values of energy cost, while the blue one denotes the values of mission efficiency, sensing accuracy and overall performance.

Number of cells. If the number of cells increases from 64 to 100, the density of drones decreases. In Fig. 4.13, *HALOP* shows a lower decrease in mission efficiency (decreased by 27.04%) than *MAPPO* (decreased by 44.50%) and *OPS* (decreased by 45.11%) as the number of cells increases. Furthermore, *HALOP* keeps relatively constant both sensing accuracy and energy cost as the number of cells varies.

Number of charging stations. If the number of charging stations increases from 4 to 9, the distance between the sensing areas and stations is reduced, cutting down the energy cost of recharging. As a result, the overall performance of both *HALOP* and *OPS* increase by 19.52% and 14.35% respectively.



(a) High density of vehicles. (b) Low density of vehicles.

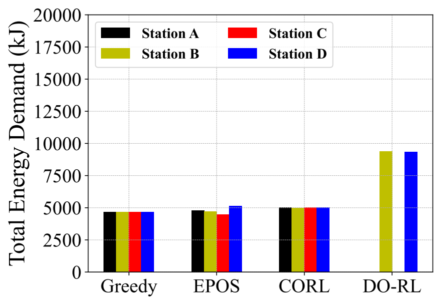
Figure 4.14: Performance comparison of the remaining battery level of four methods on both high and low density of vehicles.

Density of vehicles. If a new map area that has low density of vehicles is chosen, the distribution of sensor data changes (with different road distribution), as shown in Fig. 4.5. In Fig. 4.13, both *HALOP* and *MAPPO* have higher mission efficiency and sensing accuracy than the other two methods, with an average of 90.65% higher efficiency and 76.85% higher accuracy. This is because the learning methods observe the environment and control drones to collect data in cells and timeslots with the highest number of vehicles. *HALOP* performs better than other methods under low vehicle density even though the number of cells and charging stations increase.

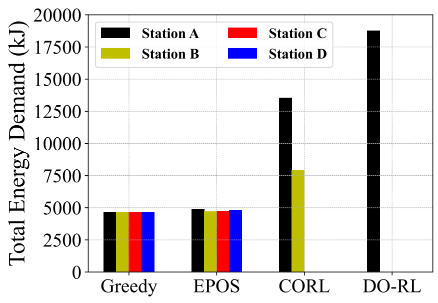
In summary, the results by varying these three parameters illustrate that *both low number of cells and high number of charging stations increase the overall performance of all methods. Long-term learning methods outperform short-term optimization methods in observing traffic flow when vehicles are sparsely distributed across the map.*

Apart from sensing performance, the battery consumption and charging of *HALOP* in complex evolving transportation scenario are studied.

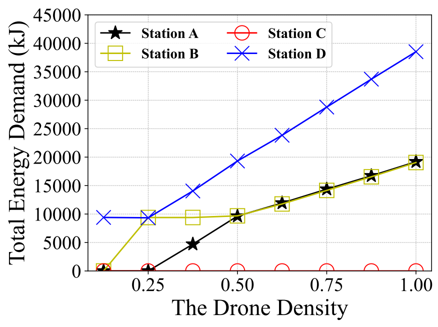
Remaining battery level. It denotes a percentage of battery level once drones have completed their sensing tasks. The value is calculated as an average among all drones within each time period. As shown in the Fig. 4.14(a) and 4.14(b), *HALOP* keeps drones at a high remaining battery level on high and low density of vehicles, with approximately 30.25% and only 1.05% lower than *Greedy-sensing*. This level follows the drones' safety regulations which suggest finishing the missions when battery life is around 25% − 30%. However, *MAPPO* does not meet the regulations under low



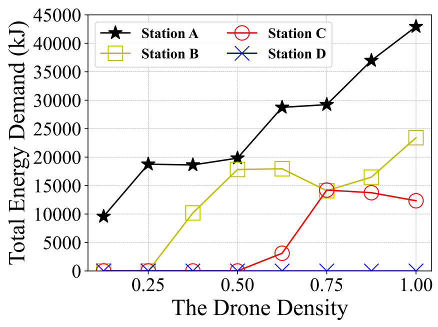
(a) Drones Density=0.25, high vehicle density.



(b) Drones Density=0.25, low vehicle density.



(c) Different drones density, high vehicle density.



(d) Different drones density, low vehicle density.

Figure 4.15: Performance comparisons of the total energy demand of four methods with 0.25 density of drones, see (a) and (b), and the *HALOP* with different density of drones, see (c) and (d).

density of vehicles, and the minimum battery level can reach 12.55%. The results illustrate that HALOP *keeps drones at a safe remaining battery level before recharging, improving drone longevity and sustainability.*

Charging load. It is the total energy demand of drones on each charging station over all time periods. This aims to study the placement of charging stations. As shown in the Fig. 4.15(a) and 4.15(b), the total energy demand on charging stations of *HALOP* over all periods is more imbalanced compared to other methods. This is because with *HALOP* drones learn to travel to the areas with the high required sensing data values. Since drones set the nearest charging stations as destinations, they rely on a lower number of charging places. For example, as shown in Fig. 4.15(c) and 4.15(d), when the density of drones is 0.125 under high density of vehicles, drones only depart from and return to the charging station D since the vehicle distribution around D is more dense than other charging stations. As the density of drones becomes 1, drones gradually rely on the charging station m and D. Similarly, drones only need to fly over the vehicle-dense areas around the charging station A and m under low density of vehicles. These results provide insights to policy makers for providing higher amount of energy on vehicle-dense areas to support drones' charging. The results show that HALOP *relies on a lower number of charging places compared to other methods.*

4.5.6 Overall comparison

Table 4.6 shows that the hybrid approach has 27.83% and 23.17% higher overall performance than standalone *OPS* and *MAPPO* respectively in the scenario of 16 drones, 8 time periods and high vehicle density. Under scarce drone resources (the low drone density), this approach is more energy-efficient and accurate by employing long-term learning methods for optimizing sensing. In contrast, under abundant drone resources (high drone density), short-term optimization methods alone suffice. Furthermore, the learning methods can direct drones towards regions with more vehicles to achieve highly efficient and accurate vehicle observation, especially in a low vehicle density area within a long time span. *HALOP* effectively combines the strengths of both short-term optimization and long-term learning methods to complement each other and improve adaptability in complex environments.

Then, this chapter evaluates the drone trajectories of all methods over time (ME is mission efficiency, SA is sensing accuracy, EC is energy cost, and MP is mean path length). Fig. 4.16 shows a visual comparison of optimal drone trajectories in *HALOP* compared to baseline methods. In this case of 16 drones fly to the grid cells and collect the required sensing data over first 4 of 8 time periods (each drone trajectory represents a color in Fig. 4.16). *OPS* fails to adapt to the changing sensing requirements and control drones to fly over the low-traffic areas, which results in higher remained amount of sensor data after period 3. As a consequence, *OPS* has lower mission efficiency and sensing accuracy than *HALOP*. Although *MAPPO* effectively learns the dynamic sensing environment through deep neutral networks, it takes a high number of unnecessary actions to cover the cells with low sensing data, leading to higher path length and higher traveling energy consumption than *HALOP*. In contrast, *HALOP*

Table 4.6: Overall performance comparison of approaches under different parameters.

Parameters / **Approaches.:**	Drone density Low (=0.25)	Drone density High (=1.0)	Time periods Low (=4)	Time periods High (=16)	Vehicle density High	Vehicle density Low
Greedy-sensing	0.14	0.62	0.16	0.16	0.15	0.16
OPS	0.22	1.78	0.20	0.28	0.23	0.23
MAPPO	0.23	1.73	0.23	0.33	0.24	0.55
HALOP	**0.28**	1.79	0.25	**0.41**	0.32	**0.70**

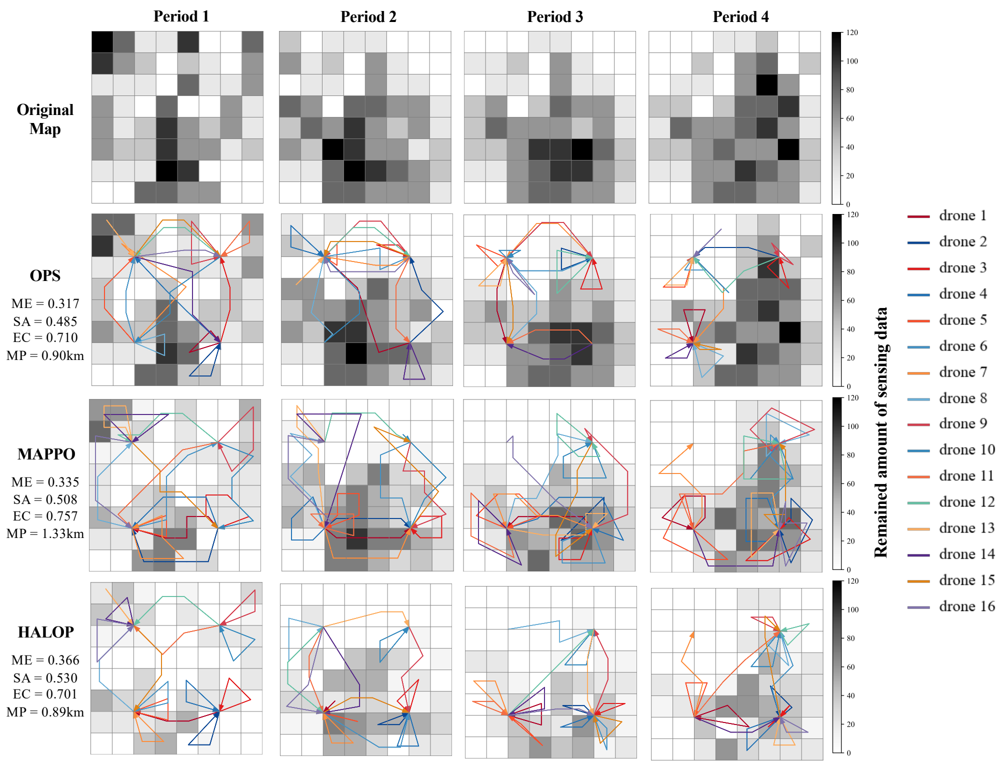


Figure 4.16: Comparison of optimal drone trajectories among all three approaches in different periods.

overcomes the drawbacks of both *OPS* and *MAPPO* by finding the high sensing areas energy-efficiently and accurately.

4.6 Comparison with Related Work

The allocation of sensing tasks within a swarm of intelligent and cooperative drones includes applications of traffic monitoring, disaster response, smart farming, last-mile delivery, etc. [5]. The UAV task allocation problem is earlier defined as a Traveling Salesmen Problem for combinatorial optimization [9, 10, 28]. An optimal task allocation for drones is found at specified places, subject to constraints including task emergency, time scheduling and flying costs. The digraph-based methods are introduced to formulate the problem of reaching optimal sensing efficiency in terms of coverage, inspection delay, events detection rate and the cost of flying trajectories. Nevertheless, existing models do not address the energy impact of task allocation since small-scale spatio-temporal scenarios do not drain the batteries of drones significantly. Although earlier work focuses on solving the energy-aware task allocation problem for large-scale and efficient sensing, it relies on a centralized system and does not consider the time of sensor data collection [102, 161, 162]. The cost of plans calculated by the total power consumption of flying and hovering [34] are modeled to optimize the distributed coordination of swarms, while accounting for the battery constraint of each drone.

In the view of the optimization approaches, scholars introduce algorithms that require a centralized computation, such as particle swarm optimization [9], genetic programming [10] and wolf pack search [28]. Chen *et al.* [28] introduce a deadlock-free algorithm to prevent two or more drones from waiting for each other in a simultaneous task, and leverage the classical interior point method and wolf pack search algorithm to solve the uncertainty problem, including the uncertainty of the flying velocity, task effectiveness and communication. Wu *et al.* [159] leverage a greedy approach by removing redundant target points from trajectories to maximize the weighted coverage while respecting energy constraints (with 5 drones to cover 225 target points). However, these centralized methods face the risk of single point of failure [163]. If the central control station fails, the task allocation process cannot recover or mitigate such failure.

Distributed task allocation algorithms include the robust decentralized task assignment [48] and the consensus-based bundle algorithm [29]. Both algorithms coordinate autonomous drones, and can ensure scalability and flexibility of multi-UAV systems to adapt to complex sensing scenarios. Nevertheless, they do not address the constraints of energy consumption. Moreover, the resilience of the algorithms is not verified, in contrast to earlier work on I-EPOS shown to preserve its learning capacity in dynamic and unstable networked environments [64]. The proposed method overcomes scalability and resilience barriers, while leaving drones with a significant level of autonomy to make a flexible cooperative selection of tasks to execute. Furthermore, the prototyping

Table 4.7: Comparison with related work in Chapter 4.

Criteria Ref.:	[159]	[48]	[29]	[26]	[99]	[101]	**This work**
Scalability to large-scale systems	✓	✗	✗	✓	✗	✗	✓
Adaptability to dynamic environments	✗	✓	✓	✓	✓	✓	✓
Sustainable long-term efficiency	✗	✗	✗	✗	✓	✓	✓
Energy-awareness	✓	✗	✗	✓	✓	✗	✓
Resilience	✗	✗	✗	✗	✗	✗	✓

problem of moving from simulation, to live deployment of decentralized socio-technical systems, and ultimately to a robust operation of a high technology readiness level as well as online iterative traffic optimization is addressed in earlier work, with I-EPOS as a case study [68, 164].

The MARL algorithm proves to be effective for addressing sensing task allocation with drones, managing complex optimization goals, energy constraints, and numerous heterogeneous agents. The MARL used by Ding *et al.* [99] for crowd sensing leverages local observation to extract the states of neighbor drones, generalized model to avoid environment nonstationarity, and PPO to prevent the destructively large policy updates. Omoniwa *et al.* [138] present a decentralized MARL approach that improves drone energy efficiency in providing wireless connectivity by sharing information with neighboring drones. Samir *et al.* [101] leverage deep deterministic policy gradient for learning the trajectories of the deployed drones to efficiently minimize the expected weighted sum age of information. However, these approaches lack autonomy for drones in self-assignment tasks, reducing adaptability to the real-time environmental changes and increasing vulnerability to agent failures [163]. Additionally, they are confined to a small-scale task allocation problem due to the training complexity.

In summary, previous approaches have either neglected the long-term advantages of strategic sensing, or failed to effectively adapt to failures and scale without encountering centralized bottlenecks. To overcome these shortcomings, the proposed approach empowers drones to make informed and proactive decisions based on predicted traffic flow, while enabling them to independently adjust to changing conditions and unexpected events, ensuring both scalability and resilience without relying on centralized control.

The comparison of related work is summarized in Table 4.7 (criteria covered ✓or not ✗). Here, *scalability to large-scale systems* refers to the ability of the method to handle increasing drone density, time periods, and environmental complexity such as the distribution of vehicles; *adaptability to dynamic environments* implies the responsiveness of the method to real-time and uncertain changes, such as the dynamic traffic flow; *sustainable long-term efficiency* denotes the capacity of the method to optimize energy resource usage over extended periods; *energy-awareness* checks if the energy consumption is considered; and *decentralization and resilience* checks if the method is decentralized and resilient.

4.7 Discussion and Future Work

In summary, **several scientific insights** on experimental results are listed as follows: (1) Under scarce drone resources, it is more efficient and accurate to employ long-term learning methods for optimizing sensing. In contrast, under abundant drone resources, short-term optimization methods alone suffice. (2) Long-term learning methods optimize drone resources usage for traffic management by directing drones towards regions with high vehicle density and advising on infrastructure planning for charging. (3) Short-term optimization aids long-term learning by simplifying agent coordination, enhancing sensing quality, and preventing energy waste. This approach ensures that drones operate efficiently while maintaining battery levels above safety thresholds, leading to more reliable and effective missions.

The proposed method can be further improved towards several research avenues, which are: (1) Use of real-world datasets in other applications of Smart Cities, including disaster response and smart farming. (2) Use of other learning method, such as multi-agent reinforcement learning algorithm, to extend the study of coordinated drones with charging capabilities and obstacle/collision avoidance. (3) Consideration of more realistic factors such as obstacles and charging capability. (4) Test other types of sensing data collection, including temperature and humidity, and study the sensor fusion that combines data from multiple sensors to create a more accurate and comprehensive understanding of an environment. (5) Use of efficient wireless communication technology and special hardware to implement the coordination capability on board and online.

4.8 Conclusions

In conclusion, this chapter illustrates the Energy-Aware Coordination of Multi-Drone Navigation and Sensing model to solve the task self-assignment problem for scalable urban sensing by a swarm of drones. To ensure energy-aware and efficient coordination of sensing, a novel plan generation strategy with three policies is introduced. Extensive experiments demonstrate that the proposed approaches of *OPS* and *HALOP* are adaptive to complex sensing scenarios and has better sensing performance than existing methods. The evaluation using realistic urban mobility provides valuable insights for the broader community: The combination of short-term and long-term strategies proves highly effective in overcoming their standalone limitations in drone-based traffic monitoring. Short-term methods compensate the challenges of training complexity, energy consumption and recharging, while long-term approaches account for maximizing accumulated rewards, enhancing the overall sensing performance. This collaborative approach allows the system to dynamically adapt to changing environments, efficiently managing varying drone resources and enabling extended operations.

Chapter 5

Coordinated Multi-Drone Last-mile Delivery

This chapter[1] investigates the task allocation problem for multi-drone systems in the context of last-mile delivery. In this scenario, drones are able to deliver parcels to doorways and balconies, avoiding human contact during pandemic and alleviating traffic congestion caused by the deliveries of heavy ground vehicles (see Section 2.4.2). Here, route planning is vital to enhance the efficiency of drone swarms in completing delivery tasks [113]. In coordinated delivery, effective route planning guides swarms of drones to reach customers in the shortest possible time or distance. Prior works model this routing problem, which is essentially a variant of the traveling salesman problem and vehicle routing problem, as a mixed-integer programming problem and uses various heuristic algorithms to solve the complex combinatorial optimization; approaches such as genetic algorithms, simulated annealing, ant colony optimization, and reinforcement learning have been widely applied [30, 114–116]. However, existing models for routing planning of drone delivery still faces two technical challenges.

The first challenge is *energy-aware planning*. Each drone must ensure its battery level remains above a minimum safety threshold (typically $25 - 30\%$) to maintain operational longevity and avoid mission failures. Moreover, energy consumption directly affects carbon emissions, which are increasingly important for achieving net zero goals [112]. While modern drones are capable of multi-parcel deliveries, existing models that solely optimize delivery time are insufficient, since they fail to account for the power consumption variation caused by parcel weight [34]. Heavier payloads lead to significantly higher energy cost. Therefore, drones must be aware of their full delivery route and select energy-optimal plans. Unlike traditional models, this chapter uses an

[1]This chapter is based on a paper on submission [23].

energy consumption model [34] that requires drones to have full knowledge of their entire route within a specific time window, allowing them to carry all parcels initially and calculate energy consumption accurately. This can be achieved by generating plans of navigation based on the designed PMAC model.

The second challenge is *delivery delay*. Delivery delay becomes a key concern when drone resources are limited and not all customer requests can be fulfilled within the maximum flight time of drones. This is especially critical in time-sensitive scenarios, such as delivering medical supplies during pandemics [110, 123]. While some existing approaches address this by prioritizing requests with the highest delivery delay (i.e., the actual arrival time minus the expected deadline) [117], it remains challenging to determine which requests to prioritize in real time. This is because customers' priority changes with time, there is a new set of customers who must be serviced at every time step. Drones know the current distribution and delivery time of customers, but do not know when and where the customers will appear in a subsequent time step. As a result, drones have to adapt to an unexpected environment in terms of the number and locations of customers and their demands. Motivated by this, the learning-based approach (*HALOP*) in PMAC model is employed to assist drones to anticipate areas of high future delay and proactively position themselves, even if that means temporarily overlooking requests that are already delayed. This 'slower is faster" strategy helps to minimize overall delay across the entire delivery horizon.

By addressing both energy-efficient planning and delay-aware prioritization, this chapter introduces the Energy-and-Delay-Aware Coordination of Multi-Drone Last-mile Delivery (EDAC-MDLD) model. It models the task allocation problem of delivery into mix-integer programming, which aims to minimize both delivery delay and energy consumption of drones while respecting their own battery and payload capacity. It also divides the problem into three sub-problems, *customer requests segmentation*, *flight range selection*, and *optimized plan selection*. First, the clustering algorithm is employed to divide the map into "service areas" for the flight range selection [165]. Second, a Multi-Agent Reinforcement Learning (MARL) is leveraged to select flight range of drones for long-term effectiveness. Third, the Optimized Plan Selection (*OPS*) is adapted based on EDAC-MDLD to find the optimal routes using a mixed-integer programming solver (e.g., Gurobi[1]). To validate the proposed algorithm, this chapter con-

[1]Available at: http://www.gurobi.com.

Table 5.1: Mathematical notations used in Chapter 5.

Notation	**Explanation**
$u, U, \mathcal{U}$	Index of a drone; total number of drones; set of drones
$n, N, \mathcal{N}$	Index of a depot node; total number of depots; set of depots
$\mathcal{V}, \mathcal{C}, c$	Set of nodes in the map; set of customer request nodes; index of a request
$t, T, \mathcal{T}$	Index of a time window; total number of time windows; set of time windows
k, K	Index of a plan; total number of plans
t_i	The expected delivery time window demanded by request i
s_c	The delivery status of request i
f_p, f_u	Function of delivery delay; function of energy consumption
m_u^{body}	Body mass of drone u
m_u^{battery}	Battery mass of drone u
m_{uij}^{parcels}	The weight of parcels carried by drone u from node j to node i
$v, \hat{v}$	Ground speed of drone; the induced velocity
d, r	Diameter of propellers; Number of propellers
θ, ϵ_u	The pitch angle of drone; the power efficiency of drone
x_{uij}	Binary variable which takes 1 if u travels from node j to node i and 0 otherwise.
d_{ij}	Traveling distance between node i and node j
E_{uij}	Energy consumption of u when it travels from node j to node i
M_u, R_u	The maximum payload of u ; the maximum flight range of u
α	The tradeoff parameter to balance delivery delay and energy consumption
π, θ^{π}	The actor network, i.e., the policy function; the parameter of the actor network
Q, θ^{Q}	The critic network, i.e., the value function; the parameter of the critic network

ducts experiments of high realism with real-world data and emulated last-mile delivery scenarios, comparing against state-of-the-art baseline methods.

This chapter is outlined as follows: Section 5.1 introduces the definitions and models used in the EDAC-MDLD model. Section 5.2 states and formulates the task allocation problem for multi-parcel delivery. Section 5.3 outlines the methodology by integrating clustering algorithm and *HALOP* to solve the problem. Section 5.4 evaluates the performance of *HALOP* under various experimental settings. Section 5.5 compared the proposed approaches with related methodologies that address last-mile delivery. Section 5.6 discusses the experimental results and outlines future work. Finally, Section 5.7 concludes this chapter.

5.1 System model

This section defines the key concepts of scenarios and the main performance metrics in the EDAC-MDLD model. Table 5.1 illustrates the list of mathematical notations.

5.1.1 Definitions and assumptions

Consider a swarm of drones $\mathcal{U} \triangleq \{1, 2, ..., U\}$ taking parcels from depots and delivering them to customer requests over a 2D map within a set of specified time windows $\mathcal{T} \triangleq \{1, 2, ..., T\}$. The whole map is partitioned into a number of service areas, each denoting a geographically bounded region, within which a drone can efficiently serve customer requests. Then, a delivery center, i.e., the depot, is set up in the center of each service area. Both depots and customer requests are the nodes distributed in the map, denoted as $\mathcal{V}$. At the time window t, the nodes can be defined as $\mathcal{V}(t) = \mathcal{N} \cup \mathcal{C}(t)$, where $\mathcal{N}$ denotes the set of depots, $\mathcal{N} \triangleq \{1, 2, ..., N\}$, and $\mathcal{C}(t)$ indicates the set of customer nodes within time window t.

To represent the traveling information of drones, the model uses the binary variable x_{uij}, which takes 1 if drone u travels from node j to node i and 0 otherwise. At each time window, drones execute the multi-parcel delivery, that is, each of them departs from a depot, deliver parcels to multiple customer requests, and returns back to the original depot or other depots that locate at the neighboring service areas. Therefore, the node j comes from the set of depots $j \in \mathcal{N}$, at the beginning of a trip, and becomes the node of customer request $j \in \mathcal{C}(t)$ after that. In contrast, the node i is the depot node $i \in \mathcal{N}$ at the end of a trip.

The set of customer requests is updated at every time window by excluding the nodes of delivered requests and including new requests in the next time window. Thus, the updating of customer requests nodes from t to $t+1$ is formulated as follows:

$$\begin{aligned} \mathcal{C}(t+1) = \Big(\mathcal{C}(t) \backslash \{k \in \mathcal{C}(t) | s_i = 0\}\Big) \\ \cup \Big(k \in \mathcal{C}(t_i) | t_i = t+1\Big), \end{aligned} \tag{5.1}$$

where $\mathcal{C}(t)$ indicates the set of customer nodes existed within time window t, $t \in \mathcal{T}$; k is the index of a customer request node; t_i denotes the expected delivery time of request i; and s_i indicates the delivery status of request i, which takes 0 if the request is delivered by a drone and 1 otherwise. It can be expressed as follows:

$$s_i = 1 - \min(\sum_{t=1}^{T} \sum_{u \in \mathcal{U}} \sum_{j \in \mathcal{V}(t)} x_{uij}(t), 1). \tag{5.2}$$

The delivery delay is also defined to measure the amount of time by which a drone arrive later than the expected delivery time t_i of a customer request, i.e., $t > t_i$. Meanwhile, if the request c is delivered ($s_c = 0$), its delivery delay is set as 0 since it is removed from the map. The model calculates the delivery delay $f_p(\cdot)$, which is formulated as follows:

$$f_p(t, t_i) = \begin{cases} (t - t_i) \cdot s_i, & t_i \leq t, t_i \in \mathcal{T} \\ 0, & otherwise \end{cases} . \tag{5.3}$$

The delivery delay prioritizes timely deliveries by penalizing late arrivals, ensuring more urgent requests are met promptly. This can lead to higher customer satisfaction and retention, ultimately enhancing long-term quality of service.

Therefore, as shown in Fig. 5.1, the overall operational flow is listed as follows (the blue arrows represent the traveling route of the drone, the red values represent the delivery delay of each request): In a time window t, each drone observes the delivery delay of undelivered requests. The delay is set as 0 at $t = 0$ but increases if the current time t is higher than the expected delivery time t_i of request, $c \in \mathcal{C}(t)$. Once the drone determines which requests it serves, it takes all required parcels initially and unloads them to these customer requests one by one. Then, the drone returns to the nearest depot where it swaps a new battery for charging and continues observing the status of customer requests at $t + 1$. The delivered requests are removed from the map, whereas the undelivered ones increase their delivery delay. After that, the drone pick-ups parcels and continues delivering at the next time window.

Furthermore, several necessary and realistic assumptions are made as follows: (1) Each drone flies at a constant ground speed and has a fixed battery capacity that limits the flight range; (2) Each customer is visited only once and by only one drone; (3) Customers who request parcels within the same time window have the same delivery delay; (4) The time of swapping batteries of drones (i.e., recharging), taking-off/landing, and loading parcels is not considered in this chapter. (5) A time window is assumed to be longer than the maximum flight time of drones. Note that this general model is also well-suited for supporting truck-drone operations: trucks act as “mobile depots”, traveling to the centers of service areas while carrying parcels and batteries for drone delivery. In addition, trucks can observe customer demand and adjust their routes by moving to other service areas as needed.

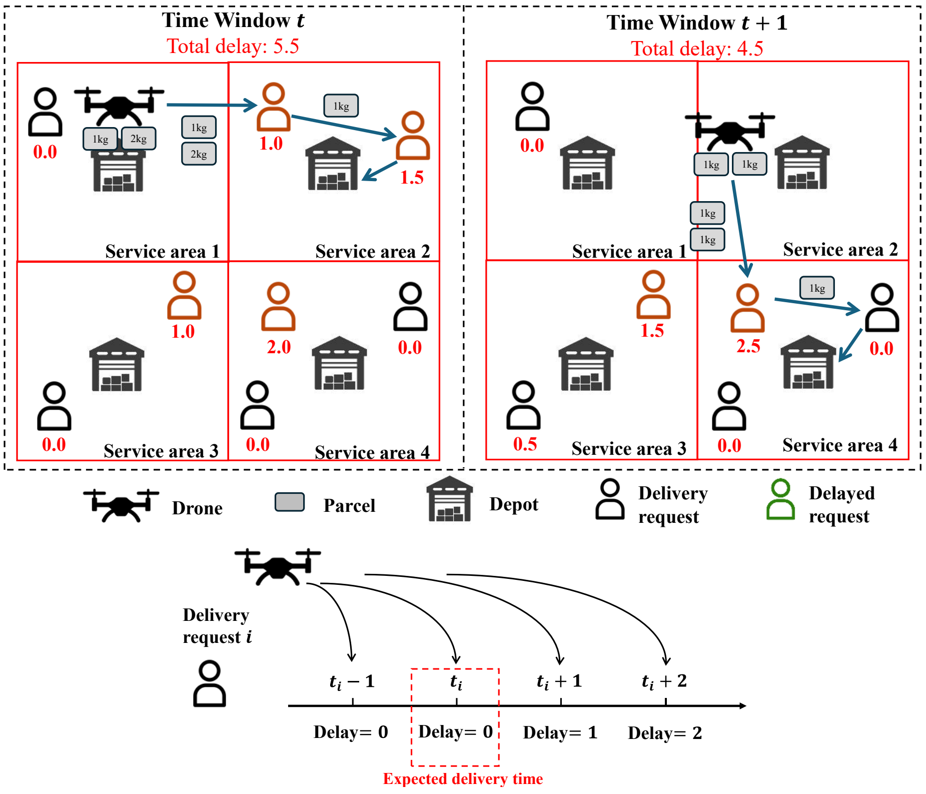


Figure 5.1: The scenario of drone delivery over three service areas within two consecutive time windows.

5.1.2 Multi-parcel energy consumption model

Energy consumption evaluates the amount of energy consumed by a drone in a trip. Drones spend energy to surpass gravity force and counter drag forces due to wind and forward motions. A drone controls the speed of each rotor to achieve the thrust and pitch necessary to hover and travel forward at the desired velocity while balancing the weight and drag forces [34]. For a drone u with mass m_u^{body} and its battery with mass m_u^{battery}, the total required thrust is formulated as follows:

$$\mathcal{T}_{uij} = m_{uij} \cdot g \cdot (1 + tan(\theta)), \tag{5.4}$$

$$m_{uij} = m_u^{\text{body}} + m_u^{\text{battery}} + m_{uij}^{\text{parcels}}, \tag{5.5}$$

$$m_{uij}^{\text{parcels}} = m_{ujk}^{\text{parcels}} - m_j^{\text{parcels}}, \tag{5.6}$$

where g is the gravitational constant; θ is the pitch angle that depends on air speed and air density; and m_{uij}^{parcels} is the weight of parcels carried in drone u from node j to node i, where $i, j \in \mathcal{V}(t)$. Eq.(5.6) implies that the drone u starts by carrying all assigned parcels and uploads each one to the corresponding customer sequentially, $k \to j \to i$, $\forall i, j, k \in \mathcal{V}(t)$. Note that drones carry no parcel to depots, i.e., $m_j^{\text{parcels}} = 0$. Given a thrust $\mathcal{T}_{uij}$, the induced velocity can be found by solving the nonlinear equation [34]:

$$\hat{v} = \frac{2 \cdot \mathcal{T}_{uij}}{\pi \cdot d^2 \cdot r \cdot \rho \cdot \sqrt{(v \cdot cos\theta)^2 + (v \cdot sin\theta + \hat{v})^2}}, \tag{5.7}$$

where v is the average ground speed; d and r are the diameter and number of drone rotors; ρ is the density of air. Therefore, given the overall power efficiency ϵ_u, the power consumption with forward velocity and forward pitch is formulated as [34]:

$$P_{uij} = (v \cdot sin\theta + \hat{v}) \cdot \frac{\mathcal{T}_{uij}}{\epsilon_u}. \tag{5.8}$$

The power consumption can be formulated as $P_{uij} = f_u(m_{uij}^{\text{parcels}})$ to represent its relationship with the weight of parcels carried by u. The energy consumption of u that travels the distance d_{ij} from node j to node i is given by:

$$E_{uij} = \frac{P_{uij} \cdot d_{ij}}{v} = f_u(m_{uij}^{\text{parcels}}) \cdot \frac{d_{uij}}{v}. \tag{5.9}$$

5.2 Problem Statement and Formulation

Based on the definitions and settings of the EDAC-MDLD model, the problem in this chapter can be stated as: *Find optimal routes of a swarm of drones $\mathcal{U}$ over all time windows $\mathcal{T}$ such that the customer requests $\mathcal{C}$ with different expected delivery time receive parcels with minimum delivery delay while reducing the energy consumption of drones.* Each route of drone u at time window t, which can be represented by x_{uij}, $i, j \in \mathcal{V}$, starts and ends at depots, passing through several customers.

The problem can be formulated as follows:

$$\min \sum_{t\in\mathcal{T}} \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{V}(t)} \sum_{j\in\mathcal{V}(t)} E_{uij} \cdot x_{uij}(t). \tag{5.10}$$

$$\min \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}(t)} f_p(t, t_i), \tag{5.11}$$

Subject to

$$\sum_{t\in\mathcal{T}} \sum_{u\in\mathcal{U}} \sum_{j\in\mathcal{V}(t)} x_{uij}(t) = 1, \forall i \in \mathcal{C}(t), \tag{5.12}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{V}(t)} x_{uij}(t) \leq 1, \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{5.13}$$

$$\sum_{j \in \mathcal{V}(t)} x_{uij}(t) = \sum_{j \in \mathcal{V}(t)} x_{uji}(t), \forall i \in \mathcal{V}(t), \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{5.14}$$

$$\sum_{i \in \mathcal{V}(t)} \sum_{j \in \mathcal{V}(t)} d_{ij} \cdot x_{uij}(t) \leq R_u, \tag{5.15}$$

$$m_{uij}^{\text{parcels}} \leq M_u. \tag{5.16}$$

The objective function (5.10) aims to minimize the total energy cost of drones. The objective function (5.11) aims to minimize the delivery delay of customers. Constraint (5.12) presents that every customer is visited exactly once. Constraint (5.13) ensures that all routes must start and end at depots. Constraint (5.14) is a connectivity constraint to ensure that each drone leaves each customer after delivery. Constraint (5.15) restricts that the maximum flight range of drone is within R_u, which covers the whole map as default. Constraint (5.16) restricts the total weight of parcels carried in the drone to be always within its maximum payload M_u.

Note that the proposed PMAC model is particularly well-suited to address this problem for several reasons. First, each drone must have complete trip information in advance, including the customer requests it must fulfill and the depot where it will land, such that it can accurately estimate energy consumption for efficient task allocation. This trip can be represented as a multi-parcel delivery route, where drones select a discrete task plan. Additionally, the problem targets at two objectives of minimizing both energy consumption and delivery delay. This forms a multi-agent combinatorial optimization problem, which PMAC is designed to solve effectively.

Second, each service area in the map must be constrained to prevent it from selecting routes with excessive energy consumption that could harm battery life. As a result, drones are limited to serving requests within neighboring service areas during each time window, as illustrated in Fig. 5.1. However, this introduces the challenge of how drones should select the appropriate neighboring service area. Here, the learning-based approach of PMAC (*HALOP*) enables drones to make strategic decisions about service area selection (or flying direction), while delegating task plan optimization to the *OPS* approach. The reinforcement learning reward function in *HALOP* is specifically designed to balance two objectives: minimizing delivery delay and reducing total energy consumption.

Moreover, the partitioning of service areas is critical, as it not only determines the size of each area, impacting the maximum energy demand on drones, but also defines the number and locations of depots. Ideally, depots should be situated in high

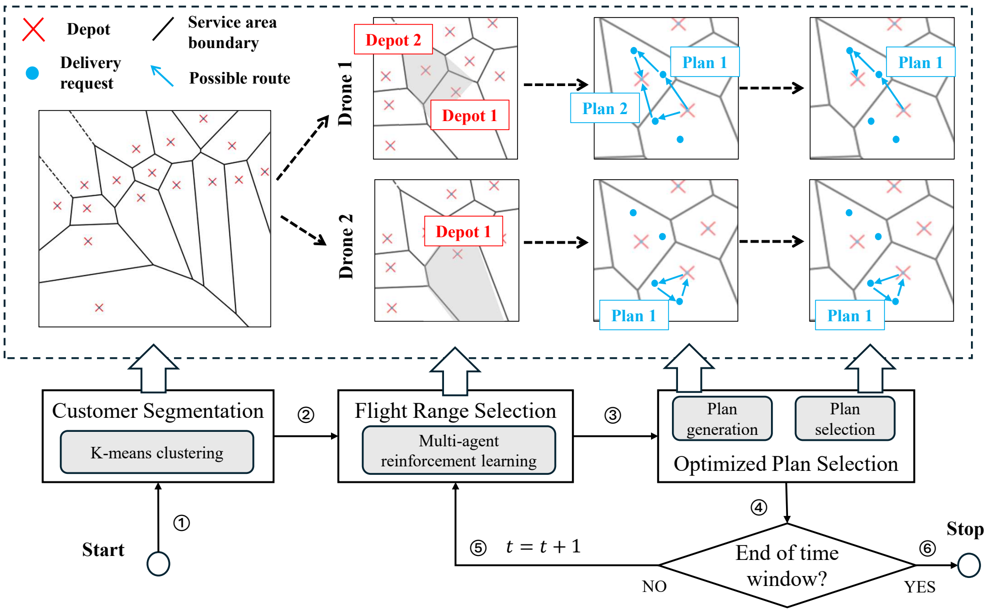


Figure 5.2: Overall framework of the methodology to solve multi-drone delivery problem.

customer-density areas to maximize service quality and operational efficiency [30].

5.3 Methodology Overview

The EDAC-MDLD model decomposes the problem into the following three sub-problems: a *customer requests segmentation* problem, a *flight range selection* problem and a *optimized plan selection* problem of drones within the flight range. In this section, the novel methodologies for each of these sub-problems are introduced.

Fig. 5.2 illustrates the overall framework of the methodology. Firstly, the K-means clustering algorithm is used for customer requests segmentation, which partitions the map into a number of service areas where the depots are located at their centers. Then, the *HALOP* approach is introduced to select a flight range of drones via reinforcement learning and solves the optimized plan selection problem of drones within the selected flight range. In flight range selection, each drone determines a flying direction that identifies the depots as the start and end points for the route (e.g., from Depot 1 to Depot 2 for drone 1, and from Depot 1 to Depot 1 for drone 2). Along with depots, their corresponding service areas must be covered by the flight range of the drone.

Algorithm 6: K-Means Clustering Algorithm.

```
1 Input: Customers dataset $\mathcal{D}$, number of depots $N$
2 while depot locations do not change do
3     Randomly select $N$ centers of service areas as depot locations
4     Calculate the Euclidean distances between depots and customers
5     Divide $\mathcal{D}$ into $N$ clusters based on the distances
6     Adjust the coordinates of depots to be the centers of corresponding clusters
7 end
8 Set the boundaries between clusters
9 Output: Set of depot nodes $\mathcal{N}$, service areas $\mathcal{N}_a$
```

Under the flight range, drones serve the customer requests and generate all possible routes, i.e., the task plans. The MIP solver selects the optimized plans for execution. Finally, drones reselect a new flight range based on their current locations and the state of customer requests in the next time window.

5.3.1 Clustering for customer requests segmentation

Before determining the flight range, the service areas with centers (i.e., the locations of depot nodes) and boundaries are obtained, which is executed before drones performing delivery. The algorithm uses K-means clustering algorithm with a historical dataset of customers that contains the information of index, X/Y coordinates, and customers' demand time. This is because K-means clustering can group the historical customer requests that are geographically close. The depots at cluster centroids are also positioned, ensuring they are centrally located relative to customer requests. This decreases the travel distance from depot to customer requests, thereby reducing the energy consumption. Additionally, K-means clustering is more computationally efficient and easy to implement in the large delivery dataset compared to other clustering algorithms such as DBSCAN and hierarchical clustering.

As shown in Algorithm 6, the proposed algorithm calculates and modifies the centers of clusters according to the Euclidean distances between the centers and customer until convergence, that is, until the centers do not change (Lines 3-6). Once the final cluster centroids are determined (serving as depot nodes), linear decision boundaries are constructed by drawing perpendicular bisectors between every pair of centroids. Each perpendicular bisector represents a boundary where all points along it are equidistant to the two corresponding depot nodes. As shown in Fig. 5.2, these boundaries–highlighted as yellow grid lines–divide the service areas and are positioned between the depot nodes, which are shown as red points.

5.3.2 Reinforcement learning for flight range selection

After finding the depot locations and the boundaries of service areas, drones begin to plan their delivery tasks within the total given time windows. At the beginning of planning, each drone u chooses a service area to determine its flight range R_u. As shown in Fig. 5.2, a drone selects a neighboring service area or the depot at which it is currently locating as the destination, and then determines the flight range that covers the corresponding service areas. This decision on flight range should consider the state of both the drone and customers, i.e., the current location of the drone and the delivery delay. Once a drone takes actions to select service areas and executes the task plan by delivering parcels to the requests, its current location is changed as it could land on a different depot. In addition, the state of customer requests updates: the delivered customer requests are removed whereas the delay time and delivery delay of undelivered customer requests increase. Therefore, the problem scenario can be represented as a Markov decision process. It can be modeled using state, action, and reward concepts:

1) *State*: The state $s(t)$ at time window t consists of four components $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$, where $\mathcal{S}_1$ represents the current location of drone u, which is the index of a service area; $\mathcal{S}_2$ reflects the current battery level of u, derived from the energy consumption model; $\mathcal{S}_3$ captures the delivery delay within the service area where u is currently located; $\mathcal{S}_4$ records the delivery delay in neighboring service areas adjacent to the location of u.

2) *Action*: The action $a(t) = \{a_1(t), ..., a_U(t)\}$ at time window t consists of the indexes of depots that are closest to the current location of drones. Each drone can choose one of the depots as its destination, which includes its departure depot. In other words, it travels to one of the neighboring service areas or remains at the previous area. Specifically, if the action has four values, then $a(t) = 0$ indicates that the drone returns to the start depot after delivering, whereas $a(t) = 1, 2, 3$ represents that the drone chooses one of three neighboring service areas.

3) *Reward*: The reward $r_u(t)$ evaluates the flight range selection of the drone. Based on the objective function in Eq.(5.11) and (5.10), the expected immediate local reward of one drone at time window t can be defined as the negative delivery delay and energy

consumption. Both two objective functions are normalized using Sigmoid. The function is formulated as:

$$\begin{aligned} r_u(t) = &-(1-\alpha) \sum_{c \in \mathcal{C}(t)} f_p(t, t_i) \\ &- \alpha \sum_{i \in \mathcal{V}(t)} \sum_{j \in \mathcal{V}(t)} E_{uij} \cdot x_{uij}(t), \end{aligned} \tag{5.17}$$

where α is a weight that makes a trade-off between the total delay of observed customers by u and the energy cost of u, $0 \leq \alpha \leq 1$. This is because these objectives often conflict: minimizing delay may require more energy-intensive operations in long route, while minimizing energy usage may neglect high-urgent customers. By adjusting the weights between 0 and 1, the operators in transportation systems can tune the reward function to prioritize or balance competing objectives based on the specific needs of the application. For example, the system can prioritize minimizing delivery delays by decreasing α to address urgent customer demands during a pandemic, or focus on reducing energy consumption and carbon emissions to meet net zero targets by increasing α.

Throughout the designed methodology, the actions of flight range selection play a pivotal role in obtaining high reward, i.e., low delivery delay and energy consumption. A “good” action balances the immediate efficiency of the flight range with the potential to serve future customers effectively, ensuring minimal delay and optimal energy usage over time. For instance, as shown in Fig. 5.1, a “good” action might direct a drone to fly to Area 2 at t, where delays are currently high. This proactive choice positions the drone to serve delayed customers more efficiently in Area 4 at $t+1$, reducing future travel. However, a “bad” action may suggest the drone remain in Area 1 to minimize immediate energy use, but at the cost of greater delays later. Thus, the proposed approach *HALOP* leverages multi-agent deep reinforcement learning to reinforce “good” actions and lower the probability to choose “bad” actions (see Section 3.4.3). It learns an optimal policy by exploring various choices of flight range, adapting to the dynamic environment and evaluating their outcomes over time.

5.3.3 Algorithm of optimized plan selection

After selecting flight ranges R_u, $u \in \mathcal{U}$, drones run the optimized plan selection algorithm to generate and select optimized task plan for delivery. The depots and customer locations within the designated service areas are modeled as a graph, where each

Algorithm 7: Optimized plan selection for multi-drone delivery.

```
Input: Parameters of drones $u$ used in Section 5.1.2, the flight range $R_u$, the maximum
       number of parcels $\hat{M}$, $\forall u \in \mathcal{U}$.
1  Initialization: Initialize a set of plans $\mathbb{P} = \emptyset$ and the sets of selected customer requests
     $\hat{\mathcal{C}_u} = \emptyset$ for each drone
2  Find $\hat{M}$ customer requests with highest delivery delay within $R_u$, and store them into $\hat{\mathcal{C}_u}$
3  Add all sets to $\hat{\mathcal{C}} = \sum_{u \in \mathcal{U}} \hat{\mathcal{C}_u}$
4  for $\forall u \in \mathcal{U}$ do
5      Find $K$ combinations of customer requests from $\hat{\mathcal{C}_u}$ under the constraint (5.16)
6      for each plan index $k := 1, ..., K$ do
7          Find the shortest path among customer requests and departure/destination depots
             via the greedy algorithm
8          Generate the plan $p_{uk}$ that includes the indexes of selected customer requests
9          Calculate the energy consumption $E_{uk}$ based upon the path as the cost of the plan
10         Add the plan and cost to the set $\widehat{\mathbb{P}}$ of delivery plans
11     end
12 end
13 Set the objective function: min $\sum_{u \in \mathcal{U}} \sum_{k \leq K} E_{uk} \cdot x_{uk}$
14 Set the constraint (1): $\sum_{k \leq K} x_{uk} = 1, \forall u \in \mathcal{U}$
15 Set the constraint (2): $\sum_{u \in \mathcal{U}} \sum_{k \leq K} \mathbb{1}\{c \in p_{uk}\} \cdot x_{uk} = 1, \forall c \in \hat{\mathcal{C}}$
16 Minimize the objective function using an optimizer tool under constraints and find the
     optimal route for each drone $p_u := p_{uk}$
Output: Optimal route $p_u$ for each drone.
```

node represents a possible stop for the drones. This algorithm contains two parts: the plan generation and plan selection, see Algorithm 7.

The plan generation part of the algorithm (Line 2-12) aims to enable each drone to generate task plans that cover the customer requests with the highest delivery delay within its flight range. Firstly, for the flight range R_u of each drone u, $\forall u \in \mathcal{U}$, the algorithm searches for a number of customer requests with the highest delivery delay within R_u and store them into a set $\hat{\mathcal{C}_u}$ (Line 2). This number is determined by the maximum number of parcels for each drone $\hat{M}$, which is empirically set according to the average weight of parcels and the payload of drones. Next, each drone u finds all combinations (equal to K) of customer requests from its corresponding $\hat{\mathcal{C}_u}$ (Line 5). The combination denotes a number of requests ($\leq \hat{M}$) selected from $\hat{\mathcal{C}_u}$ whereas their total weight is no higher than the drone payload M_u. Then, each drone finds the shortest path for each combination of customer requests, calculates the energy consumption

E_{uk}, and generate the plan p_{uk} (Line 6-11). All plans and their corresponding cost (i.e., energy consumption) are stored into the set $\mathbb{P}$.

In the plan selection part (Line 13-16), the objective function is set to minimize the total cost of plans selected by drones, where x_{uk} denotes binary variable which takes 1 if drone u selects the plan l and 0 otherwise. Two constraints are then applied: (1) each drone must select exactly one plan, and (2) each customer request must be visited exactly once across all selected plans, where $\mathbb{1}\{c \in p_{uk}\}$ is an indicator function which takes 1 if $c \in p_{uk}$ and 0 otherwise. Once the constraints are set, the optimal route for each drone is then obtained, yielding the most cost-effective delivery plan. Finally, the algorithm selects an optimal route for each drone using an optimizer tool such as Gurobi [6].

The overall training process of the methodology primarily involves the following steps (see Algorithm 8): At the beginning of each episode, the environment is reset with the input of delivery data, which includes the coordinates and time of customer requests. The state of drones are also initialized. Next, the algorithm runs the K-means clustering in Algorithm 6 for customer requests segmentation. Then, each agent (or drone) takes an action to select a service area based on its current state. These actions also determine the departure and destination of drones, i.e., the start and end depots. After generating and selecting a task plan for delivery through Algorithm 7, drones calculate their immediate reward and transition to a new state. The buffer, a data storage structure used for experience replay, stores all transitions of each drone. Several groups of transitions are sampled randomly (H groups of transitions) for updating the parameters of both the critic and actor networks. Finally, the algorithm updates the parameters in critic network and actor network (see Section 3.4.3).

5.4 Experimental Evaluation

In this section, an overview of the simulation settings is presented. The delivery dataset, specification of drones, and the used neural networks are introduced. Then, the baselines and performance evaluation metrics are discussed. Finally, the results are assessed across various scenarios.

Algorithm 8: The *HALOP* training for multi-drone delivery.

```
Randomly initialize critic network $Q(\cdot)$, actor network $\pi(\cdot)$ with parameters $\theta^Q$, $\theta^\pi$
for episode := 1 to max-episode-number do
    Reset the customer requests and the state of drones
    Partition the map and set the locations of depots via Algorithm 6
    for period $t$ := 1 to max-episode-length do
        for $\forall u \in \mathcal{U}$ do
            Take action: $A_t^u = \pi(S_t^u|\theta^\pi)$
            Find the destination depot and the flight range $R_u$
        end
        Generate task plans and find the optimal one for each drone via Algorithm 7
        for $\forall u \in \mathcal{U}$ do
            Compute reward and update state
            Store transition sample into buffer
            Sample a random mini-batch of $H$ samples from buffer
        end
    end
    Estimate advantage via Eq.(3.18)
    Calculate the probability ratio via Eq.(3.19)
    Update $\theta^\pi$ by minimizing the loss via Eq.(3.20) (3.21)
    Update $\theta^Q$ by minimizing the loss via Eq.(3.22)
end
```

5.4.1 Experimental settings

The experiment models a real-world delivery scenario in Shanghai City based on a last-mile delivery dataset is modeled, named LaDe[1]. LaDe is a large-scale dataset consisting of data, such as time and location, for 10,677,000 packages collected by 21,000 couriers in 5 cities across 6 months. The dataset comprises data for both pickup and delivery of packages for 30 regions in Shanghai City.

As shown in Fig. 5.3, this study selects an area of $10 \times 10km$ in the city where a number of customers request parcels at different times from 9 am to 5 pm. It illustrates an example of the distribution of customer requests on August 20, 2022 (shown in black points). The left picture shows the distribution of 16 depots (white crosses) and the boundaries of service areas (white lines) with indexes (white number) determined by K-means clustering based on LaDe dataset. The right one shows the depots and

[1]Available at: https://huggingface.co/datasets/Cainiao-AI/LaDe.

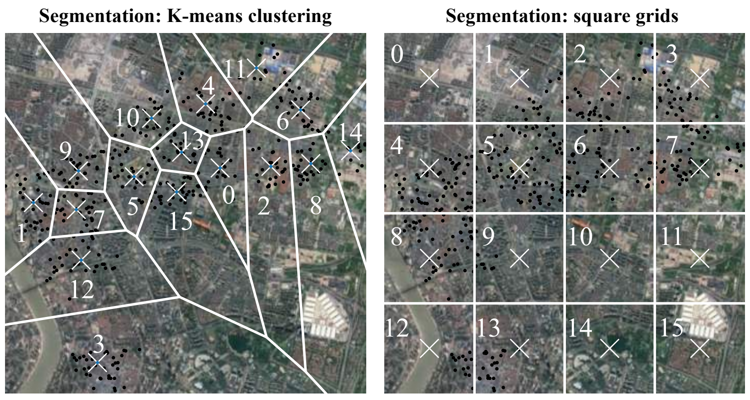


Figure 5.3: Customer requests segmentation of a city map with the area of $10 \times 10km$ in Shanghai City.

Table 5.2: Notations for delivery drones.

Notation	**Value**
Mass of drone body	$m_b = 2kg$
Mass of battery	$m_e = 1kg$
Diameter of propellers	$d = 0.5m$
Number of propellers	$r = 4$
Ground speed	$v = 10m/s$
Power efficiency	$\epsilon_u = 0.8$
Battery capacity	$C^u = 160kJ$

boundaries determined by square grids, which will be used for baseline comparison in Section 5.4.2. The total duration is divided into 12 time windows, each representing 30 min, when a drone executes a delivery mission and fly back to a depot. Except for X/Y coordinates, each request has an actual delivery time in the dataset, which is assumed to be the expected delivery time. The cross-validation is employed for 90 days of data: 80% for training and 20% for testing.

Furthermore, this study refers to the parameters of Meituan drone because of its high flight range and payload [166]. Each drone has a maximum flight range of around 3 km and a maximum payload of around 2.5 kg. The other drone parameters used in this chapter for the simulations are set as listed in the Table 5.2.

In the settings of neutral network and learning algorithm, a total of $H = 64$ groups of transitions are sampled as mini-batches in a replay buffer, with a discount factor of

$\gamma = 0.95$ and a clip interval hyperparameter of 0.2 for policy updating by using proximal policy optimization [135]. The algorithm uses the recurrent neural network with 64 neurons in the two hidden layers in both critic and actor networks. The activation function used for the networks is tanh. The models are trained over $\mathcal{E} = 10,000$ episodes, each consisting of multiple epochs (equal to the number of time windows).

5.4.2 Baselines and metrics

A direct comparison between the proposed methods and existing work is challenging due to the limited relevant work. Most existing approaches are not well-suited for multi-objective combinatorial optimization problems that involve both time-sensitive constraints and multi-parcel delivery requirements. For this reason, this chapter compares the proposed method of Hierarchical Learning-based Plan Selection (*HALOP*) with the following relevant methods:

- *OPS-global*: The flight range of each drone is set as the default, that is, every drone searches the customer requests over the whole map within each time window regardless of its flight range. It does not solve customer requests segmentation and flight range selection.

- *OPS-random*: K-means clustering algorithm is leveraged to determine the service areas; drones select their sub-ranges randomly. *OPS-random* does not support any long-term strategic navigation between depots.

- *MAPPO*: It is a state-of-the-art MARL algorithm using proximal policy optimization [135]. Different from *HALOP* that takes actions to select a flight range, the action of this method is to choose a task plan from the set $\hat{\mathcal{C}_u}$ without the help of Gurobi optimizer.

- *HALOP-square*: It leverages MARL-based flight range selection and optimized plan selection algorithms, but divides the map into square grids as service areas for customer requests segmentation instead of K-means clustering.

All the algorithms are evaluated using the following two key metrics: (1) *Mean energy consumption*, which denotes the mean energy consumption per drone that travels within all time window based on the objective function (5.10); (2) *Average delivery*

Table 5.3: Performance comparison of five methods on the basic delivery scenario.

Metrics	*OPS-global*	*OPS-random*	*MAPPO*	*HALOP-square*	*HALOP*
Mean energy consumption (kJ)	1551 ± 106	662.0 ± 40.4	664.3 ± 33.2	1768 ± 126.7	669.1 ± 33.2
Average delivery delay (hour)	0.67 ± 0.20	1.17 ± 0.16	1.46 ± 0.10	1.15 ± 0.11	0.88 ± 0.10
Combined cost	0.66 ± 0.09	0.57 ± 0.06	0.67 ± 0.04	0.87 ± 0.07	$\mathbf{0.48 \pm 0.04}$
Delay unfairness	0.21 ± 0.01	0.60 ± 0.02	0.34 ± 0.02	0.56 ± 0.08	0.41 ± 0.01
Average running time (second)	6.23 ± 0.11	0.28 ± 0.04	0.38 ± 0.02	0.46 ± 0.02	0.46 ± 0.02
Average earlier arrival time (hour)	0.03 ± 0.06	0.52 ± 0.17	1.03 ± 0.05	0.11 ± 0.04	0.17 ± 0.06

delay, which indicates the average delay time (hours) per customer request all time windows based on the objective function (5.11);

Moreover, several other metrics can also be observed on these algorithms: (1) *Combined cost*, which is the weighted sum (using trade-off weight α) of normalized mean energy consumption and average delivery delay. (2) *Delay unfairness*, which measures the delivery delay inequality among search areas. Specifically, the total delivery delay of customer requests in each service area is calculated, and the unfairness value between these total delivery delay values is calculated using Gini coefficient. (3) *Average running time*, which calculates the average time (seconds) it takes these algorithms to run per repetition. (4) *Average early arrival time*, which measures the average amount of time (hours) by which drones arrive earlier than the expected delivery time (or deadline) across all customer requests over all time windows. (5) *Depot load*, which quantifies the total weight (kg) of the parcels picked up by the drones from each depot over all time windows.

5.4.3 Evaluation on basic delivery scenario

The *basic delivery scenario* is evaluated at first. There are 16 service areas with 8 dispatched drones. The number of actions is 4, which indicates that each drone selects one of the four nearest depots as the destination (including its departure depot). The tradeoff parameter is set as 0.5 as default. The purpose is to compare the performance of different approaches.

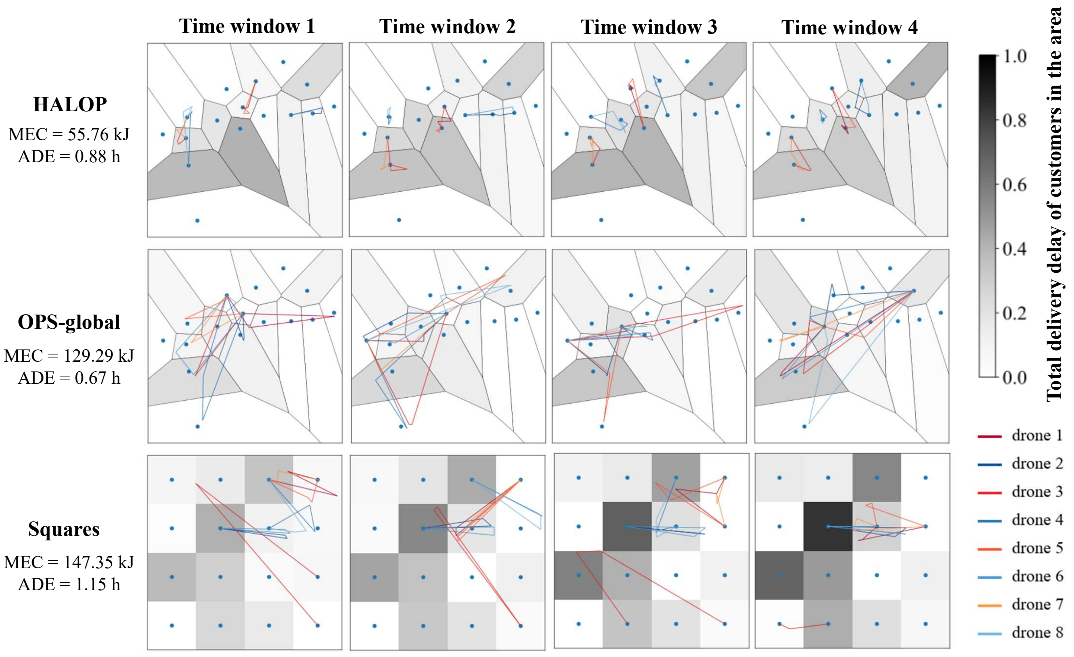


Figure 5.4: Comparison of optimal drone trajectories among all three approaches in different time windows (MEC is mean energy consumption, ADE is average delivery delay).

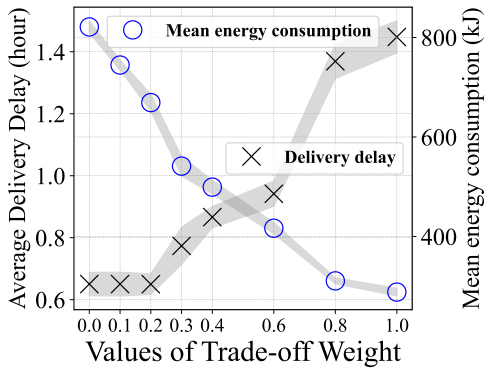
Table 5.3 shows the metric performance of all methods which run 40 times to obtain average values and standard deviation errors. Among all methods, *OPS-global* has the lowest average delivery delay and delay unfairness, but incurs the highest average running time due to its extensive search range. When the search range is reduced, *HALOP* can run approximately 93.73% faster than *OPS-global*, making it more suitable to real-time application. *OPS-global* also has the higher mean energy consumption than *HALOP* by 882.4kJ since drones are more likely to choose distant customers regardless of their maximum flying time. The comparison of drone trajectories is illustrated in Fig 5.4. Each drone takes around 80.80% of battery capacity on average, exceeding its safe battery level. Based on the average carbon intensity in the United Kingdom in 2024, which is 125g CO_2 per kWh [167], *HALOP* can reduce approximately 245g CO_2 less than *OPS-global*. In overall, *HALOP* has 27.27% lower combined cost than *OPS-global*, validating the energy-efficiency of the proposed approach.

Except for *OPS-global*, *HALOP* has lower delivery delay than other methods, by controlling drones to strategically target high-delay areas while still maintaining low energy consumption. While *OPS-random* minimizes the mean energy consumption, it lacks a long-term strategy that guides drones to the customer requests with the highest delivery delay. As a result, drones using *OPS-random* arrive earlier than those using *HALOP* by an average of 0.35 hour, but they also experience longer delays, averaging 0.29 hour more. This random selection of flight range also leads to unequal serving of customer requests, with 0.19 higher than *HALOP*. Even though *MAPPO* prevents drones from choosing incorrect service areas, it performs the worst in terms of delivery delay, with the highest average delivery delay and earlier arrival time. This is due to inefficient exploration in training caused by high action space. In addition, *HALOP* using K-means clustering significantly outperforms *HALOP-square* using square grids with a 1099kJ reduction in mean energy consumption and 0.27 hour decrease in average delivery delay. This is because the K-means clustering restrict drones to fly over high request density areas, whereas square grids may result in that drones waste their energy to choose a corner grid without any delivery requests, see Fig. 5.4.

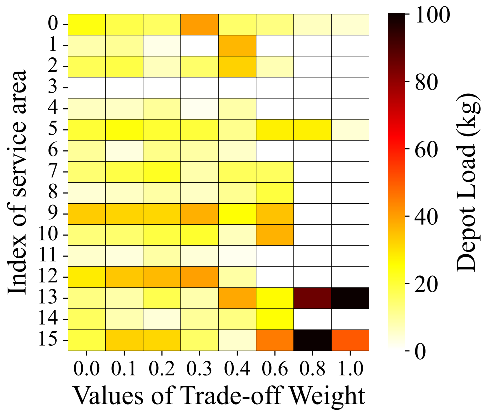
5.4.4 Evaluation on complex delivery scenario

The *complex delivery scenario* is then studied by varying three parameters: (1) The trade-off weight to balance the delivery delay and average energy cost defined in the reward function; (2) the number of drones used in the delivery mission; and (3) the flying coverage density to represent the ratio of the number of areas a drone chooses to fly over the total number of service areas in the map. The goal is to assess the scalability of the proposed approaches under different experimental conditions. Due to the least performance of *MAPPO* and *HALOP-square* in basic delivery scenario, the methods *HALOP* with *OPS-global* and *OPS-random* are compared in the complex synthesis scenario.

Trade-off weight. Fig. 5.5(a) illustrates the influence of trade-off weight on the balance between mean energy consumption and average delivery delay of *HALOP*. The shadow represents the error (i.e., standard deviation) of metrics. As the value of trade-off weight α increases, the energy consumption of each drone decreases linearly while the delivery delay of customers increases significantly when $\alpha > 0.2$. The heavier trade-off weight on the energy cost restricts drones to travel less and gradually choose to stay at the service areas (see Fig. 5.5(b)). As a result, drones fail to deliver distant customers request with high delay. When $\alpha \geq 0.6$, drones only pick-up parcels from area 0, 5,



(a) Trade-off weight vs. delivery delay & average energy cost.



(b) Trade-off weight vs. depot load in each service area.

Figure 5.5: Performance comparison of *HALOP* across different values of trade-off weight.

13 and 15, see Fig. 5.3. *HALOP* only produces 80.04g CO_2 at the cost of 1.45 hour average delivery delay when $\alpha = 1$. In contrast, lower α reduces the delivery delay by controlling drones to fly across all areas and take parcels from their corresponding depots. *HALOP* achieves a relatively equal average delivery delay with *OPS-global* when $\alpha = 0$, but 730.3kJ lower mean energy consumption. The results highlight that HALOP *can achieve either low mean energy consumption or low average delivery delay by adjusting the tradeoff parameter. The high cost of energy consumption comes with benefits on timely delivery and vice versa.*

Number of drones. As shown in Fig. 5.6, if the number of drones increases, the average delivery delay of all methods decreases as drones can cover a wider area and deliver to customer requests on time. This proves that sufficient drone resources can minimize the arriving time of all customer deliveries and efficiently mitigate the customer delay. The effect is significant in *OPS-global* which has has 0 delay with more than 16 drones and the lowest delay unfairness, albeit with high energy consumption and running time. When the number increases to 20, *HALOP* has the lower mean energy consumption by approximately 850.0kJ and average running time by around 88.8 seconds compared to *OPS-global*, leading to 21.95% lower combined cost. This demonstrates the applicability of the proposed approach in energy-constrained real-time delivery scenarios. Moreover,

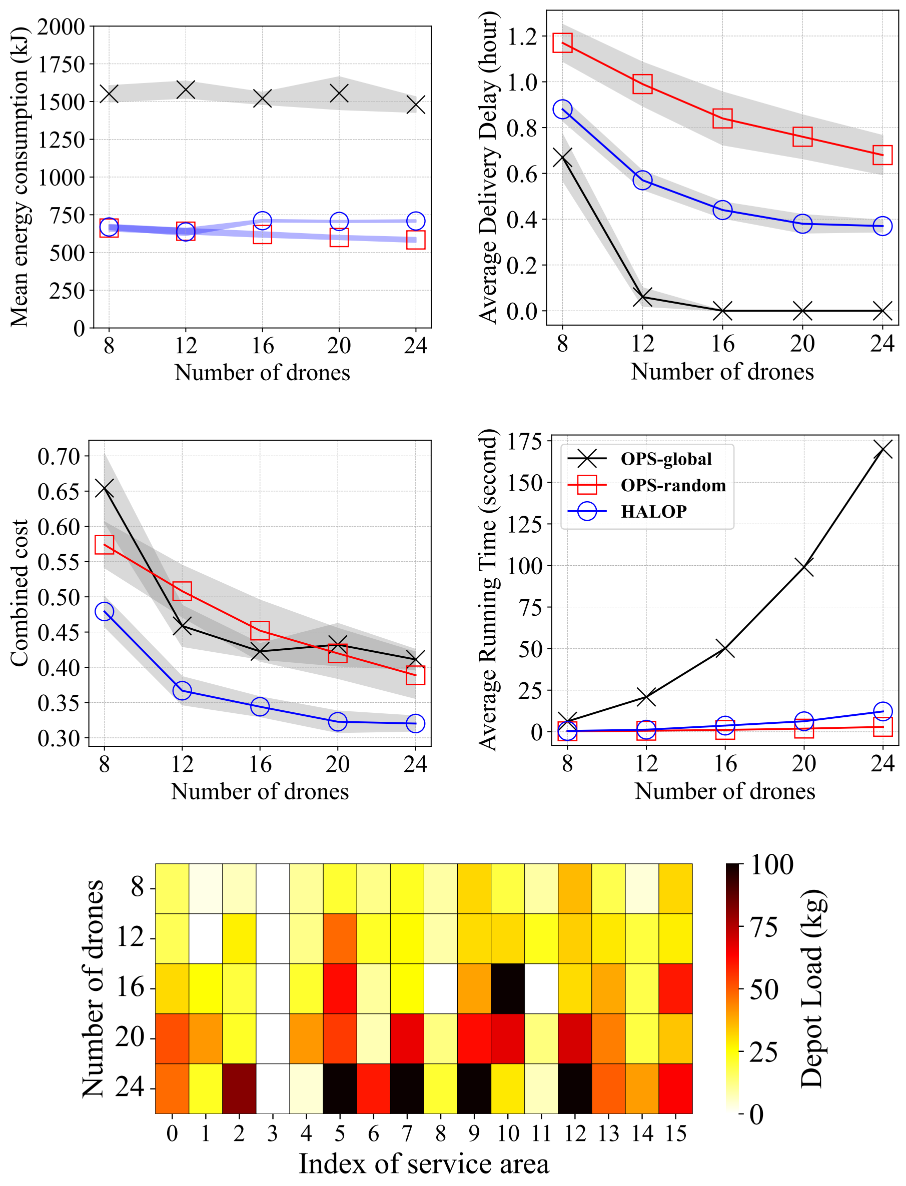


Figure 5.6: Performance comparison of three methods in all five metrics across different number of drones.

HALOP learns the flying strategy efficiently, decreasing the delivery delay by approximately 24.25% compared to *OPS-random*. In terms of the depot load, drones are gradually required to take parcels from all except the depots in service areas 3 and 11 (see Fig. 5.3), as the number of drones increases. This is because these low-load depots

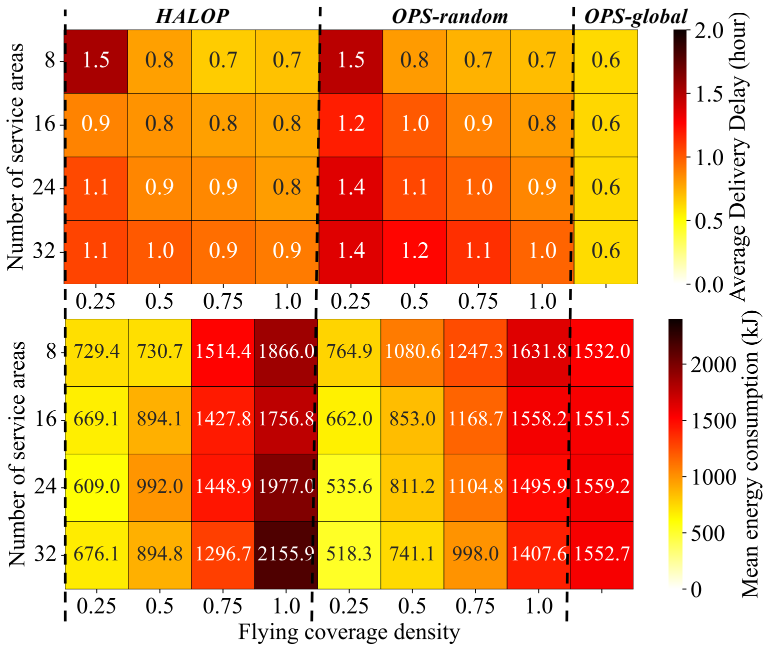


Figure 5.7: Performance comparison of three approaches across different flying coverage density and number of service areas.

are located at the borders of the map and are hard to access. The results show that *high number of drones decreases average delivery delay of* HALOP, *while maintaining a low mean energy consumption and a low running time.*

Flying coverage density. When the flying coverage density increases from 0.25 to 1.0, the number of service areas a drone chooses to fly increases by fixing the total number of service areas. Fig. 5.7 illustrates the performance of *HALOP* and baseline methods when using varying flying coverage densities and numbers of service areas. On the one hand, if the flying coverage density increases from 0.25 to 1.0, the flight range of both *HALOP* and *OPS-random* extends, which helps drones to reach customer requests with higher delivery delays, thereby reducing delivery delay of the system. As the coverage density rises, the mean energy consumption increases due to a greater likelihood to choose longer flight routes. Note that in the case of high flying coverage densities, *HALOP* typically selects only a single flight range, rather than covering the entire map. This narrower range can make it more challenging to find optimal routes compared to *OPS-global.* On the other hand, if the number of service areas increases from 16 to 32, while fixing flying coverage density, *HALOP* maintains mean energy consumption with only an 8.96% variance but achieves a 12.52% reduction in average delivery delay compared to *OPS-random.* It is important to note that too few coverage options can hinder learning efficiency, as seen in the scenario with 8 service areas at 0.25 density. The results show that HALOP *searches customers and reduces delivery delay more energy-efficiently than* OPS-random*, which operates with a fixed and low flying coverage density.*

5.5 Comparison with Related Work

The proposed work differs from the existing literature in the following aspects. First, existing methods to overcome the limitations of the range and payload capacity of drones have been to pair them with delivery trucks such that drones only perform deliveries within small ranges ("last-mile deliveries") [120–122, 168, 169]. Furthermore, the use of multiple depots [169, 170] and recharging (in-flight or with distributed charging stations) [171–173] has been proposed previously. However, synchronization between drones and trucks is a critical and complex problem with significant costs due to idle times [121]. These solutions are also not environmentally friendly; drones typically have fewer CO_2 emissions than delivery trucks, and life-cycle impacts are increased if additional depots are required [174, 175]. Moreover, drone deliveries without trucks become cost-competitive when more packages can be carried per tour [176]. Only a few models have explored swarm-based *multi-parcel drone delivery* scenarios where numerous drones operate together to make *multiple* deliveries directly from the depot [120, 177, 178], or from a truck [179, 180]. This work differs from the dominant truck–drone models by studying a more universal scenario where drones carry multiple packages directly from a depot and deliver packages to customers based on the urgency of their delivery.

Second, this chapter performs *delay-aware drone deliveries* in which a customer's delivery priority increases with the waiting time; in other words, it is more urgent to deliver packages to customers who have had to wait longer for the deliveries. In contrast, most of the literature so far has considered priority that remain constant through the entire service period [37, 178, 181]. Wang et al. [181], for example, employs fuzzy theory to deal with demand arrival rate under a priority queuing strategy, but their priority queue remained constant throughout the operational period. Similarly, Narayanan et al. [178] considers service delays by using a system of prioritization to compensate the customers differently for late deliveries; however, they also used a fixed priority queue.

Third, most of the papers in the literature consider energy consumption as a fixed limit on drone flight time or flight range [120, 182]. It is only recently that some related research has considered the battery energy consumption rate [178, 183, 184]. The proposed approach in this chapter considers the battery energy consumption rate with multiple package deliveries. When a package is delivered, the weight of the drone decreases, thereby reducing the energy consumption rate and increasing the range. Unlike most prior literature that consider a single package and use a direct relationship between consumption and weight, this chapter explicitly accounts for this increase with the number of packages. Furthermore, very few studies have considered both delay awareness and energy consumption together. Some studies have examined the tradeoff between energy consumption and delivery time (e.g., [185, 186]), but they consider fixed delivery times, essentially applying the capacitated vehicle routing problem with time windows to drone deliveries.

Finally, the proposed approach is different from most drone delivery systems that typically consider static environments; for example, previous work [187] proposes Deep-ETA, a spatial-temporal sequential neural network model for estimating travel (arrival) time for parcel delivery, formulated with multiple destinations. However, this approach relies on the availability of information regarding delivery routes. In Deng at al. [188], the authors consider travel time uncertainty and the minimization of service delay risks (with a truck-and-drone delivery system), but the drones had complete information. Recent research that has employed reinforcement learning approaches, e.g., [119, 189, 190] focuses on relearning flight paths and delivery strategies, assuming complete information of all the customers and all possible routes. In contrast, the proposed approach is adaptive to new customers and locations that change with increasing demand urgency at different times. This is achieved by using MARL to improve optimization; MARL is used to observe new delivery demand and take strategic policies to visit the customers with the most urgent priority. This makes the proposed approach highly adaptive to various operational scenarios and conditions.

Table 5.4 compares the most prominent recent literature (criteria covered ✓or not ✗). Similar to the comparison table in Section 4.6, *scalability to large-scale systems* refers to the ability of the method to handle increasing numbers of customers, drones, tasks, and environmental complexity; *adaptability to dynamic environments* refers to

Table 5.4: Comparison with related work in Chapter 5.

Criteria Ref.:	[180]	[181]	[186]	[188]	[185]	[184]	[178]	**This work**
Scalability to large-scale systems	✗	✗	✗	✗	✗	✗	✓	✓
Adaptability to dynamic environments	✓	✓	✗	✗	✓	✗	✗	✓
Sustainable long-term efficiency	✓	✗	✗	✗	✓	✓	✗	✓
Energy-awareness	✓	✗	✗	✓	✓	✓	✓	✓
Delay-awareness	✗	✓	✗	✗	✗	✗	✗	✓
Multi-parcel	✓	✗	✗	✗	✗	✗	✗	✓
No. of Customers	25–100	0–50	15–36	10–25	10–100	5–50	10–10000	400 – 600

the responsiveness of the method to real-time and uncertain changes, such as the distribution of customer requests; *sustainable long-term efficiency* refers to the capacity of the method to optimize resource usage, such as energy and time, over extended periods; *energy-awareness* implies that energy consumption is considered; *delay-awareness* checks if delivery delays have been considered; and *multi-parcel* refers to whether a drone can carry multiple packages.

5.6 Discussion and Future Work

In summary, several new insights are obtained from experimental results that are summarized as follows: (1) The proposed optimized plan selection (OPS) algorithm effectively coordinates energy-aware drones to minimize delivery delays, while supporting energy-efficient multi-parcel delivery. (2) Strategic selection of flight range using reinforcement learning and clustering algorithms in $HALOP$ ensures low delivery delay and effectively prevents drones from searching high distant customer requests. It not only reduces energy consumption and carbon emissions of drones, but enhances operation speed, making it highly adaptive to real-time delivery applications. (3) The tradeoff between energy consumption of drones and delivery delay of customer requests allows operators in transportation systems to make appropriate adjustments according to their priorities, such as net zero sustainability and pandemic delivery. (4) Experimental results on depot deployment optimization, addressing aspects such as the number, location, and parcel load distribution of depots, support service providers with infrastructure planning, thereby enhancing overall logistics efficiency and adaptability to dynamic customer requests.

In the future work, a large number heterogeneous drones with different batteries, payloads, and power efficiencies will be considered. This will aim to support the large-scale delivery that requires the decentralized combinatorial optimization approaches, such as collective learning [18, 178]. Furthermore, this work needs to conduct experiments on a real-time drone testbed to validate the applicability of the proposed method.

5.7 Conclusions

In conclusion, this chapter illustrates the Energy-and-Delay-Aware Coordination of Multi-Drone Last-mile Delivery model to solve the last-mile delivery problem by a swarm of drones, which aims to minimize both energy consumption of drones and delivery delay of customer requests. Due to the constraints of flight range and parcel weight, the problem is modeled by a multi-objective combinatorial optimization problem using a multi-parcel energy consumption model. This problem is then decomposed into three sub-problems, which are addressed by three approaches respectively: (1) The clustering algorithm for customer requests segmentation; (2) The reinforcement learning approach for flight range selection; (3) The optimized plan selection algorithm for delivery navigation. A synthesis of these approaches is proposed to improve the long-term delivery efficiency by learning the policy on selecting an optimal flight range using *HALOP* proposed in the designed PMAC model. Extensive experimentation using a real-world delivery dataset from Shanghai city provides valuable insights for sustainable, efficient drone delivery. The results demonstrate that *HALOP* not only makes a tradeoff and minimizes both energy consumption (or carbon emissions) and delivery delay, but shows rapid operational speed and recommends depot deployment, making it adaptive for real-world logistics.

Chapter 6

Multi-drone Testbed Prototyping

This chapter[1] aims to validate the applicability and realism of the PMAC model based on the experimentation testbed using hardware drones. Designing, prototyping, deploying, testing, and evaluating task allocation solutions (spatio-temporal sensing or last-mile delivery) is a highly complex and interdisciplinary research endeavor. On the one hand, simulation environments reduce the complexity and the number of influential environmental variables, allowing in this way a more targeted study of task allocation algorithms. One the other hand, experimenting with real drones in indoor and even outdoor environments increases realism and external validity [191].

To bridge this gap, this chapter introduces a testbed to study multi-drone coordination for task allocation problems. The testbed consists of a 2D area of interest (a map). For instance, consider image/video capturing. Such an area can be simply implemented by one large or several smaller interconnected monitors lying on the floor in an indoor lab environment, a projector or another visual medium. These means can project images, videos and more sophisticated visual content of simulation models (e.g. traffic flows [164]) to emulate the tasking environment. Low-cost drones can traverse the map and hover to "scan" points of interest from a certain height, or carry payload (equivalent to a parcel) and deliver it to the points, which are regarded as the warehouses or customers. As such, different points may have different task requirements encoding in this way a large spectrum of missions and applications. In this simple and generic context, the proposed testbed allows the study of high Technology Readiness Level (TRL) solutions for a plethora of problems including: charging control of drones, collision avoidance, sensor fusion, delivery, autonomous search and navigation, optimiz-

[1]This chapter is based on two published papers [26, 27].

ation, learning and task allocation algorithms (e.g., multi-agent collective learning and reinforcement learning), among others.

However, the indoor testbed still lacks the realism of outdoor environments, particularly regarding in-flight safety issues such as obstacle-to-drone and drone-to-drone collisions. Designing and prototyping a testbed that integrates collision avoidance in task allocation for multi-drone coordination is a complex challenge. On the one hand, task allocation enables autonomous and flexible coordination for efficient missions of sensing and delivery, but increases the risk of unpredictable collisions, requiring highly sophisticated algorithms for collision detection and avoidance, especially in small lab spaces. On the other hand, incorporating collision avoidance introduces new hardware testing challenges, as it alters the navigation and tasking outputs by task allocation, impacting factors such as energy consumption and task accuracy.

To tackle these challenges, a *Multi-drone Tasking Experimentation Testbed* (M-TET) is designed to improve the realism of multi-drone task allocation operations. This testbed studies various task allocation problems in drones, including charging control, navigation, and collision avoidance. As a proof-of-concept, an artificial potential field algorithm [36] is applied to predict collision fields and determine optimal drone flight paths, effectively detecting and mitigating potential collisions. This method ensures collision-free navigation and tasking, which are further coordinated and optimized using the proposed PMAC. Extensive experimentation with low-cost real drones, an indoor tasking environment, and real-world traffic data from Athens [4] validates the effectiveness of M-TET in traffic vehicle monitoring, demonstrating its capacity to move complex task allocation and collision avoidance algorithms for drones to real-world.

The organization of this chapter is outlined as follows: Section 6.1 introduces the design of indoor multi-drone testbed M-TET. Section 6.2 illustrates the prototyping details of M-TET, including drones, tasking environments, task allocation and collision avoidance. Section 6.3 evaluates the experimental results from the hardware tests. Section 6.4 compares M-TET with related testbeds and experimentation. Section 6.5 discusses the new insights on experimental results of M-TET and outlines future work. Finally, Section 6.6 concludes this chapter.

6.1 Testbed Design

M-TET is the first prototype of an indoor drone testbed with the aim to support interdisciplinary research on multi-drone coordination, including: energy-aware learning and optimization algorithms for task allocation, control and communication problems of different drones as well as sensing/delivery scenarios for different applications of Smart Cities. The proposed testbed relies on a model, which can be implemented in different lab environments. At an abstract level, the testbed is modeled by the elements presented in the rest of this section.

6.1.1 Elements description

Interactive drones. They communicate to interact with each other directly, or via low-latency edge proxies, or through the cloud [154]. Each drone can run its task allocation and collision avoidance software for sensing and delivery within the following continuum [68]: (i) offline/online, remote, centralized computations (server deployment), (ii) offline/online, remote, distributed computations (edge-to-cloud deployment scenario) [154], and (iii) online, locally on drones, distributed computations. For long-term missions, each drones can support wireless charging and be fully charged before starting the next missions [192].

Indoor tasking environment. It is the area that drones perform different types of tasks, e.g., camera recording for spatio-temporal sensing, and weight carrying for last-mile delivery. In the sensing scenario, drones sense at multiple resolutions determined by the height of hovering or flight. At each resolution (height), the map is split into cells, each defining the sensing area of interest. The higher the height, the larger the cell and the lower the sensing quality are likely to be, e.g., image recognition, sound sensitivity, etc. In the context of a sensing mission, each cell has specific sensing requirements that determine the hovering duration and data acquisition of UAVs. For instance, areas with high traffic may also have higher sensing requirements to accurately capture the traffic flows with drones. In an indoor lab environment, a sensing map can be emulated with one or more monitors or a projector illustrating images, videos and visual outputs of simulations models, e.g. a traffic flow simulation with SUMO [164]. Furthermore, the map can be divided into a number of square cells or set multiple target points, which will be regarded as the warehouses for multi-drone delivery. Each drone with a weight

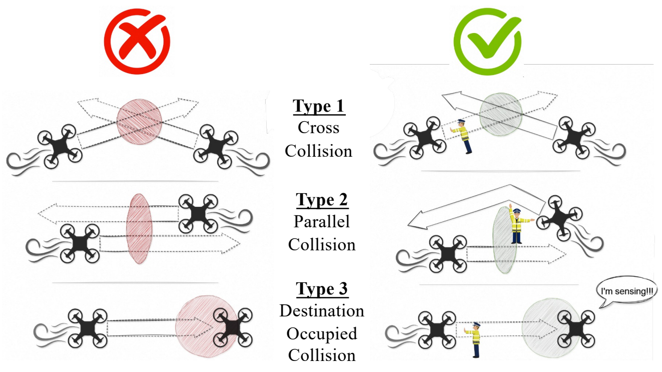
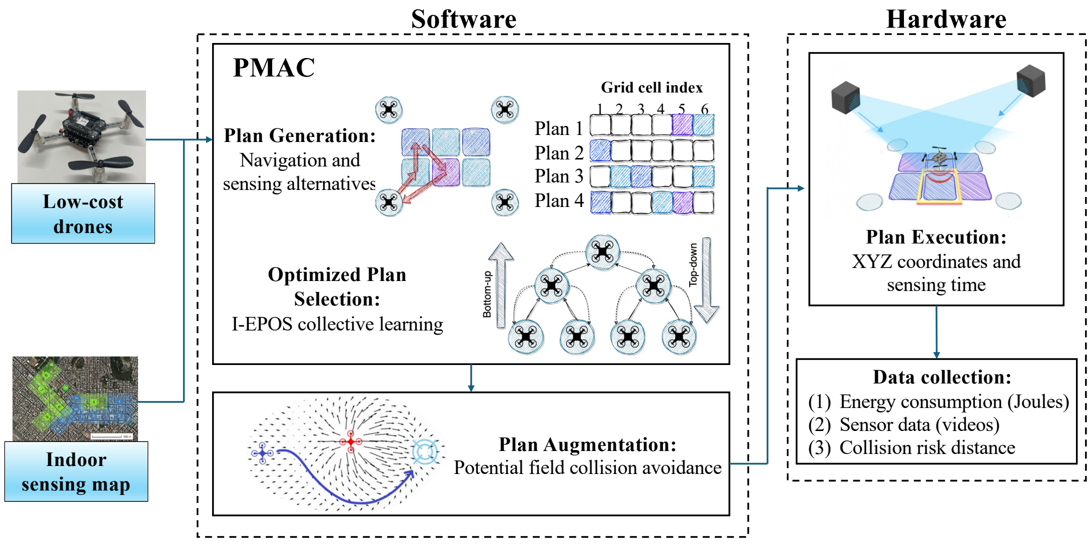


Figure 6.1: Three types of collisions and corresponding avoidance methods.

attached flies to a point of interest and lands, delivering to the required warehouse. In this context, each cell/target has specific delivery requirements (i.e., a binary value) that whether it needs to be delivered at each time period. In this way, task allocation algorithms can be assessed at low-cost, with ease and safety in a wide range of what-if scenarios.

Task allocation algorithm. It assists multiple drones to plan the navigation and tasking in a coordinated way such that each selects one plan influenced by the selections of others. Each position index of the plan denotes the area of interest in the map, such as the cell. Therefore, the global plan, i.e., the aggregated plans selected by the swarm matches well the task requirements of the environment. This matching represents the relative approximation between the total actual values per cell and task requirements per cell. Error and correlation metrics such as the root mean squared error, cross-correlation or residuals of summed squares can estimate this matching [39]. To solve the task allocation problem, the proposed approaches in PMAC are used in this chapter.

Collision detection and collision avoidance. It detects all possible in-flight and static collisions, i.e., intersections of flight paths or walls in multi-drone missions, and then minimizes the likelihood of collisions during the path planning. This requires an intelligent path planning algorithm, making task allocation cost-effective and safe within various scenarios. The algorithm detects intersections of flight paths and obstacles, and augments the navigation and tasking plans selected via task allocation to prevent the potential collisions, minimizing in-flight risks, i.e., the traveling distance at high risk of collisions. Fig 6.1 illustrates three typical types of collisions considered in this chapter



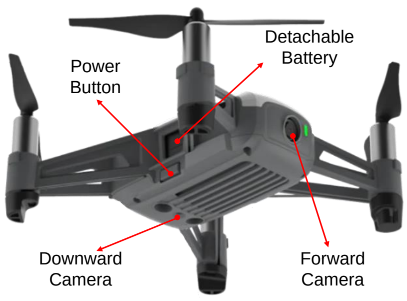
Software
Hardware
PMAC
Low-cost drones
Indoor sensing map
Plan Generation: Navigation and sensing alternatives
Grid cell index
1 2 3 4 5 6
Plan 1
Plan 2
Plan 3
Plan 4
Optimized Plan Selection: I-EPOS collective learning
Bottom-up
Top-down
Plan Augmentation: Potential field collision avoidance
Plan Execution: XYZ coordinates and sensing time
Data collection:
(1) Energy consumption (Joules)
(2) Sensor data (videos)
(3) Collision risk distance

Figure 6.2: An overview of the prototyped M-TET architecture.

(cross, parallel, and destination-occupied). Cross collision denotes two drones fly across each other; Parallel collision indicates two drones fly towards each other; Destination-occupied collision means one drone performing sensing or delivery occupies another drone's destination.

6.1.2 Architecture overview

Fig. 6.2 illustrates an overview of M-TET architecture. The core of M-TET lies in two software approaches. One is the designed PMAC model, which divides in plan generation and plan selection parts. It generates for each agent a finite number of discrete navigation and tasking options, each with an estimated power consumption: the *possible plans* and their *cost* respectively. Plan generation is performed using the drones specifications (weight, propeller and battery parameters), and the tasking environment (wind and grid cells). Then the agents run multi-agent collective learning of I-EPOS [18] to make a selection such that all choices together add up to maximize the task accuracy (matching). However, these selected plans have no information about the potential collisions, and thus the path planning algorithm for collision avoidance is required to augment selected plans. This testbed uses the artificial potential field algorithm [36] to generate attractive forces towards the destinations and repulsive forces between drones. These forces guide the drones to their target cells for tasking according



(a) Drone specification.

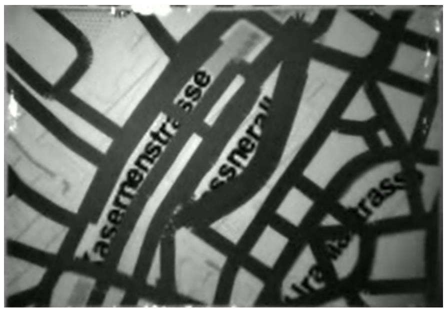
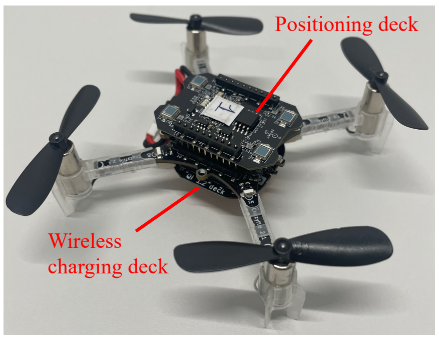
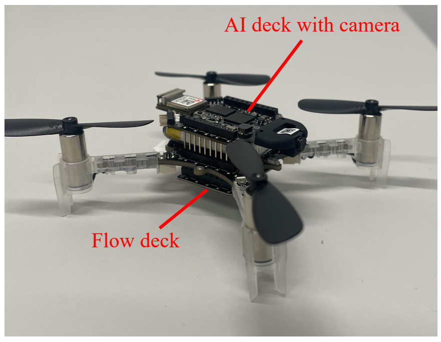

(b) Camera image cell.

Figure 6.3: The DJI Tello EDU drone that flies and hovers to capture images of cells in the sensing map.



(a) Crazyflie for testing positioning and wireless charging.



(b) Crazyflie for navigation and camera recording.

Figure 6.4: Assembly of Crazyflies for two types of functions.

to the selected plan, while also avoiding potential collisions. Finally, the algorithm produces the output plans containing the X/Y coordinates duration at each cell. They are then executed by drones in hardware, which records the energy consumption, the collected sensor data and collision risk distance during the mission. Therein, collision risk distance implies the specific distance at which the risk of collisions becomes significant, illustrated by the path length within the shaded area shown in Fig. 6.1.

6.2 Testbed Prototyping

This section introduces the prototyping details of M-TET architecture.

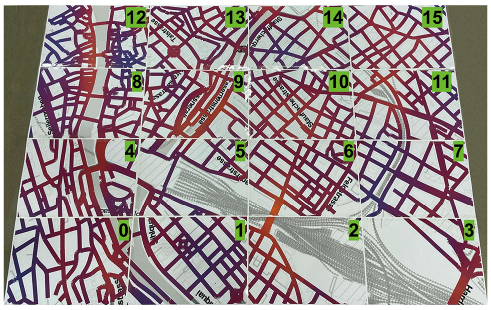
6.2.1 Hardware drones for tasking

M-TET uses two types of hardware drones for experiments: the DJI Tello EDU UAV and the Crazyflie 2.1.

DJI Tello EDU UAV. The DJI Tello EDU UAV[1] is used because of its size, weight, and accessibility [193]. This drone can be programmed in Python, Swift or Scratch, and multiple DJI Tello UAVs can fly for swarm applications. It has 7 min flight time and precise hovering mode. The drone has 0.1kg weight, 7.26cm propeller length, a battery capacity of $1100mA \cdot h$ and it is configured to fly with an average ground speed of 0.1m/s. Based on this information, both the hovering and maneuvering power can be estimated. DJI Tello UAV has a semi-open-source structure, i.e., the UAV cameras can be accessed by sending commands code, but its internal structure cannot be changed. This drone has 2 different cameras: forward and downward as shown in Fig. 6.3(a). The forward camera can record RGB 5 mega-pixel and 1280×720 videos, whereas the downward camera, which is used to capture images, can only record gray-scale and 320×240 video (see the cell in Fig. 6.3(b). Moreover, DJI Tello UAV can handle small payloads up to approximately 80 grams, making it feasible to test delivery tasks.

Crazyflie 2.1. The Crazyflie 2.1 is used because of its size, weight and accessibility[2]. The drone has 27g weight, 47mm propeller length, a battery capacity of 250mAh (LiPo battery) and it is configured to fly with an average ground speed of 0.1m/s. Based on this information, the power consumption can be estimated. Crazyflie can be programmed in Python with the support of API in Bitcraze[3]. This is further expanded by the Crazyswarm API[4], which provides additional functionality to control a swarm of Crazyflies. The drone can be mounted by multiple hardware decks to support different functions, such as positioning (with the support of lighthouse base stations), camera recording, and wireless charging. Due to maximum weight limitation of Crazyflie, two types of drones are assembled, one is for testing wireless charging, see Fig. 6.4(a), and the other is for collecting sensor data, see Fig. 6.4(b). The data-collection drone with an ultra low power 320×320 grayscale camera is equipped with a tiny mirror to take videos of ground.

[1]https://www.ryzerobotics.com/tello-edu
[2]https://www.bitcraze.io/products/crazyflie-2-1/
[3]https://github.com/bitcraze/crazyflie-lib-python
[4]https://github.com/USC-ACTLab/crazyswarm



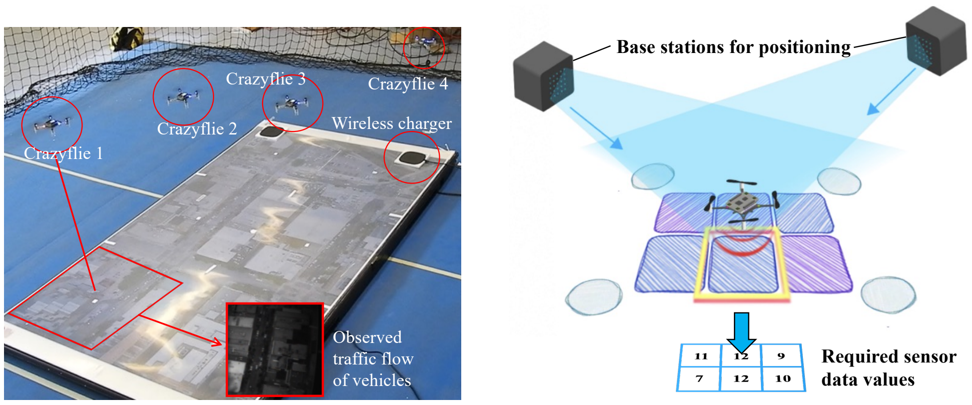
(a) Test printout map.

60	60	60	60
60	60	60	60
60	0	60	60
60	0	0	0

(b) Sensing requirements.

Figure 6.5: Test environment made by a printout for drone tasking.

The DJI Tello EDU UAV offers the advantage of easy integration for tasks (e.g., camera recording and parcel delivery) and supports multiple functions simultaneously. However, it lacks a robust API for swarm navigation and coordination. In contrast, Crazyflie 2.1 provides powerful tools for controlling drone swarms but faces challenges in testing multiple functions at once, such as navigation, tasking, recharging, and collision avoidance, due to its limited weight and battery life. To leverage the strengths of both hardware drones, this testbed employs the DJI Tello EDU UAV to evaluate individual drone navigation and tasking along predefined routes, while multiple Crazyflie drones are used to study task allocation and collision avoidance behaviors.


Crazyflie 3
Crazyflie 4
Crazyflie 2
Wireless charger
Crazyflie 1
Observed traffic flow of vehicles
Base stations for positioning
11
12
9
7
12
10
Required sensor data values

(a) A 75 inch screen to display the traffic flow of vehicles.

(b) Grid cells in the video and sensing requirements.

Figure 6.6: indoor tasking lab using a large screen, Crazyflies, wireless chargers, and lighthouse base stations for positioning.

6.2.2 Setting up an indoor tasking environment

To investigate both short-term and long-term vehicle observation, two types of indoor tasking environment are considered: (1) A printout map that consists of a number of square cells as the areas/points of interests for sensing and delivery; and (2) A sensing video to show the traffic flow of vehicles over a long time span.

Printout map. A printout map of size 168×118 centimeters is tested consisting of $4 \times 4 = 16$ cells as shown in Fig. 6.5(a). The map is made by A3 printouts (one per cell) that when put together they show a 2D map of an area in Zurich (see Fig.4 in earlier work [149]). The cell with index 0 is the departure/landing cell. For the delivery tasks, the central point of each cell in the map (see Fig. 6.5(a)) is assumed to be a warehouse from which customers can pick up the parcels. The delivery requirements are simplified by setting each cell with a binary value, that is, the value is 1 if the warehouse in the cell is required to deliver, and 0 otherwise. For the sensing tasks, each cell can be sensed effectively from a height of $40cm$ based on the field of view that the camera of the drone has. The map is augmented by another layer depicting a continuous kernel density estimation of cycling risk calculated by past bike accident data and other information [149]. This overlay map can determine the sensing requirements (hovering time) as follows:

$$t(n) = \frac{f_R(n)}{\sum_{n=0}^{N-1} f_R(n)} \cdot T, \tag{6.1}$$

where n $(0 \leq n \leq N-1)$ denotes the index of a cell on the map out of a total of N cells, $f_R(n)$ is the kernel density estimate and T is the maximum total operating time of all drones. This means that more data are collected from cells with higher cycling risk, emulating in this way a cycling safety application scenario. For testing, the sensing requirements are simplified by setting each cell with a value of 60 when it contains cycling/road infrastructure, and with 0 when it is does not, see Fig. 6.5(b).

Sensing video. A 75 inch screen is set on the ground as an indoor environment to emulate the outdoor sensing environments. As shown in Fig. 6.6(a) and Fig. 6.6(b), the screen is divided into 2x3 square grid cells, each with an area of 55x47cm. Each cell can be sensed effectively from an altitude of 50cm (i.e., the vehicles can be clearly observed). Other accessory equipment is set, including lighthouse base stations and wireless chargers. To show the significant and broad impact of M-TET on a transportation scenario, the screen displays the video of the traffic flow of vehicles recorded from satellites or other simulators, e.g., Simulations of Urban Mobility (SUMO). Thus, drones fly to grid cells and collect sensor data by recording the videos of traffic vehicles. In this scenario, a higher sensing requirement in a cell represents a high urgency for traffic monitoring over this area, which requires drones to hover a longer time to measure accurately. M-TET uses a real-world dataset of vehicle trajectories named pNEUMA[1] collected by a swarm of drones in the congested downtown area of Athens, Greece [4].

6.2.3 Task allocation using collective learning

Due to the restricted flight time and the limited number of drones, M-TET faces difficulties in testing a long-term tasking. Thus, this chapter uses the *Optimized Plan Selection* (OPS) approach in PMAC to address the task allocation problem.

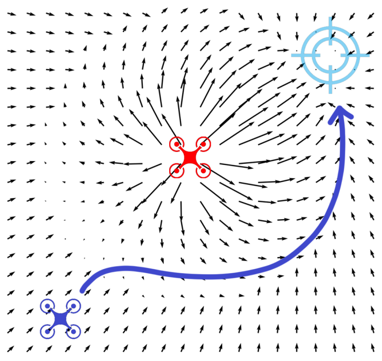
A number of N agents, each corresponding to a drone, autonomously generate 16 plans, each comprising sequences of M real values representing sensing duration or delivery assignment at cells. These plans, serving as random samples of alternative routes with a random number of cells, respect battery constraints. They are assigned costs based on the power consumption of flying and hovering [34].

The plan selection in *OPS* is made in a coordinated way using multi-agent collective learning (I-EPOS) [18]. For this, agents connect into a balanced binary tree topology within which they interact with their children and parent to improve iteratively their plan selection. The goal of the agents is to minimize the root mean square error (RMSE) between the following signals: values per cell summed up over all agents and the task requirements per cell. The agents perform 40 bottom-up and top-down learning iterations. For this testbed prototype, the optimization process is performed offline and remotely, however deployments of I-EPOS for online optimization are already available for future extensions [68].

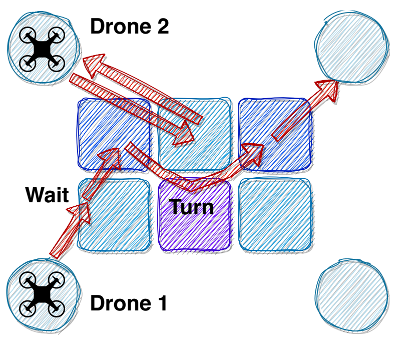
6.2.4 Collision avoidance using artificial potential field

M-TET applies an artificial potential field algorithm [36] to path planning of drones, which creates force to repel drones from obstacles and attract them towards their destinations. As shown in Fig. 6.7, a Potential Fields Grid (PFG) in this implementation is created for each drone (named as the target drone) per timestamp. It is a 2D-grid of vectors, where each vector points in the direction that the target drone should fly

[1]https://open-traffic.epfl.ch/



(a) Layout of potential field grids to the target drone.



(b) An example of collision avoidance with 2 drones.

Figure 6.7: An example of collision avoidance using artificial potential field.

at that position per timestamp [36]. In Fig. 6.7(a), the blue drone is the target drone attracted by the destination, whereas the red drone is the obstacle drone repelling the target drone. The vectors influenced by both attractive and repulsive forces point towards the navigation of the target drone. In Fig. 6.7(b), since drone 2 has higher priority and stronger repulsive force than drone 1, drone 1 is pushed out of the target cell and “wait” until drone 2 passes by, and makes a “turn” when traveling to the next cell.

There are two types of PFG: attractive PFG and repulsive PFG, which coexist to balance the drone navigation. The attractive PFG generates forces that pull the drone towards its target destination. The repulsive PFG creates forces that push the drone away from obstacles (e.g., the other drones, walls and other static obstacles). These repulsive forces are stronger when the target drone is closer to an obstacle. After the summation of all types of forces, the drone navigates towards its destination while avoiding obstacles. Thus, given I vectors, a vector with index of i at timestamp t, $i \geq 1$, $t \geq 1$, consists of two components: attractive component V^{a} and repulsive component V^{r}, formulated as follows:

$$V_i(t) = \sum_{j \in \mathcal{O}} V^{\mathrm{r}}_{i,j}(t) + V^{\mathrm{a}}_i(t), \tag{6.2}$$

where the repulsive one is effected by the obstacle j, $j \in \mathcal{O}$, where $\mathcal{O}$ defines the set of obstacles in the map; the attractive component is influenced by the current destination for the target drone n (this destination is changed once the drone reaches it). The attractive component can be formulated as:

$$V_i^{\mathrm{a}}(t) = V_i^{\mathrm{a}}(t-1). \tag{6.3}$$

However, there is a the common problem with repulsive PFG: agents stand-off in situations where they continuously repel each other without making progress, leading to a deadlock scenario [36]. To mitigate the lock, the priorities of drones are assigned randomly. Drones with higher priority exert stronger and more extensive repulsive forces, pushing lower-priority drones out of their paths. As a result, drones can reach their respective destinations in sequence without getting obstructed by obstacles. The maximum radius of repulsion effect of a drone is defined as follows:

$$R_j = D^{\mathrm{min}}(1 + lnP_j), \tag{6.4}$$

where P_j denotes the priority index of the obstacle j. The higher value of P_j is, the higher priority of an obstacle drone is (P_j is a constant if j is a static obstacle like walls). D^{min} denotes the minimum distance between a drone and an obstacle before they collide. This chapter sets 25cm considering the wind force caused by the Crazyfly. Thus, the update of a vector in repulsive PFG $V_{i,j}^{\mathrm{r}}$ with index of i per timestamp t is formulated as follows:

$$V_{i,j}^{\mathrm{r}}(t) = \begin{cases} \frac{V_{i,j}^{\mathrm{r}}(t-1)\cdot S_j^2}{|V_{i,j}^{\mathrm{r}}(t-1)|\cdot D_{i,j}(t)}, & D_{i,j}(t) \le R_j \\ 0, & otherwise \end{cases}, \tag{6.5}$$

$$S_j = \delta|V_i^{\mathrm{a}}(t-1)| + lnP_j, \tag{6.6}$$

where $D_{i,j}$ indicates the distance between the vector i and an obstacle j; S_j denotes the scaling factor for the strength of repulsion, higher than the strength (or magnitude) of attractive component of the vector V_i^{a}. This ensures that the repulsive force acting on the target drone is stronger than attractive force, thus influencing the vector V_i according to Eq.(6.2). The scaling value is set $\delta = 2.5$ empirically to ensure that the lowest priority drone is strong enough to repel the other drones under the attractive forces. Therefore, the vectors in PFG prioritize maintaining a safe distance and prevent potential collisions over reaching the destinations. For normalization, the magnitude of both repulsive and attractive components is set as $|V_i^{\mathrm{r}}(1)| = |V_i^{\mathrm{a}}(1)| = 1$. Finally, the target drone is forced by the summation of all vectors per timestamp, formulated as $\sum_{i=1}^{I} V_i(t)$ for I vectors.

6.3 Experimental Evaluation

To evaluate M-TET, this chapter focuses on the sensing missions of traffic monitoring using camera-equipped drones for city road surveillance. The primary objective is to manage the spatio-temporal flight behaviors of a drone swarm to accurately observe vehicles and traffic flow, aiding in the detection of congestion and accidents. In this section, the baselines and performance evaluation metrics are illustrated at first. Then, evaluation on energy, sensing and collision avoidance are made using the testbed scenario and complex simulation scenarios.

6.3.1 Experimental settings

The approach used in M-TET is the collective learning of I-EPOS based on potential field collision avoidance, named as *EPOS-PF*. To assess the collision avoidance, two baseline approaches are introduced: collective learning without collision avoidance (*EPOS*) and collective learning with custom collision-based scheduler (*EPOS-CA*). *EPOS-CA* considers three classical types of collisions during the in-flight missions of drones as shown in Fig. 6.1. These collisions are detected after sensing plans are selected and avoided by delaying drones with lower priority. Cross and destination-occupied collisions are mitigated by controlling drones to wait until the path is collision-free while parallel collision is removed by redirecting drones to a point away from its original path before it resumes back to its target cell. To compare with *EPOS* that minimizes RMSE to improve task accuracy regardless of energy consumption, this chapter introduces another baseline method that agents can make a choice that minimizes the energy consumption of their drones while preventing collisions using artificial potential field, named as *Greedy-PF*.

The evaluation of all approaches includes key metrics:

Energy consumption. It is the total energy consumed by all drones, calculated by their hovering and traveling time.

Risk of collisions. It represents the ratio of the total travel distance where drones are at risk of collision. It can be calculated by d_r/d, where d denotes the total traveling distance of drones, d_r indicates the collision risk distance.

Sensing mismatch. It denotes the RMSE between the total sensing of drones (i.e., the number of observed vehicles) per cell and the sensing requirements (i.e., the number

Table 6.1: Results of optimizing sensing for each DJI Tello UAV.

UAV Index	**Battery Level** (%)			**Visited Cells Indices**						**Total Time** (s)	**Actual Power** (w)	**Hovering Power** (w)	**Maneuvering Power** (w)
	Start	End	Diff.	1st	2nd	3rd	4th	5th	6th				
1	75	47	28	0	7	10	12	14	15	143.35	30.80	31.80	31.92
2	86	69	17	0	4	6	7	11	12	135.37	30.98	31.80	31.92
3	99	62	37	7	9	10	12	14	15	159.32	31.42	31.80	31.92
4	100	77	23	4	9	11	12	13	-	135.31	31.59	31.80	31.92
5	80	55	25	4	6	7	9	10	12	150.70	30.36	31.80	31.92
6	100	68	32	4	6	7	8	9	11	152.95	31.50	31.80	31.92
7	88	75	13	0	4	8	13	14	15	130.91	30.89	31.80	31.92
8	100	76	24	0	8	13	14	15	-	125.50	31.59	31.80	31.92
9	100	74	26	0	6	8	10	11	13	136.22	31.59	31.80	31.92
10	100	63	37	8	9	10	11	14	15	153.29	31.42	31.80	31.92

of vehicles acquired from pNEUMA [4]) per cell.

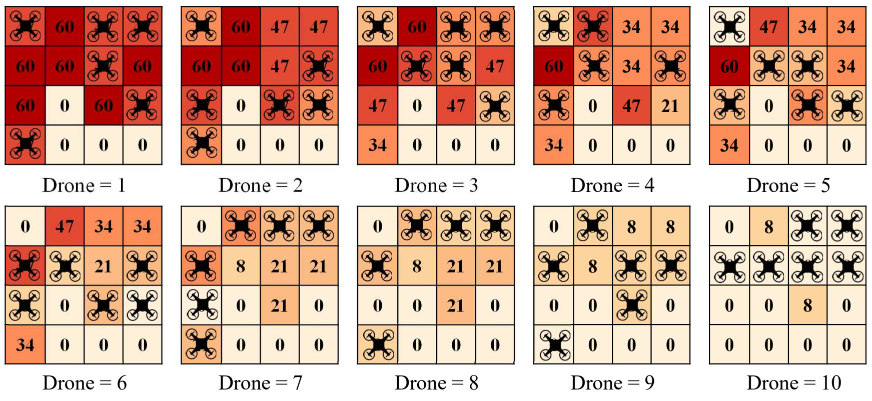
6.3.2 Evaluation on energy and sensing

The accuracy of energy consumption and the task accuracy are firstly evaluated using DJI Tello UAVs and the printout map.

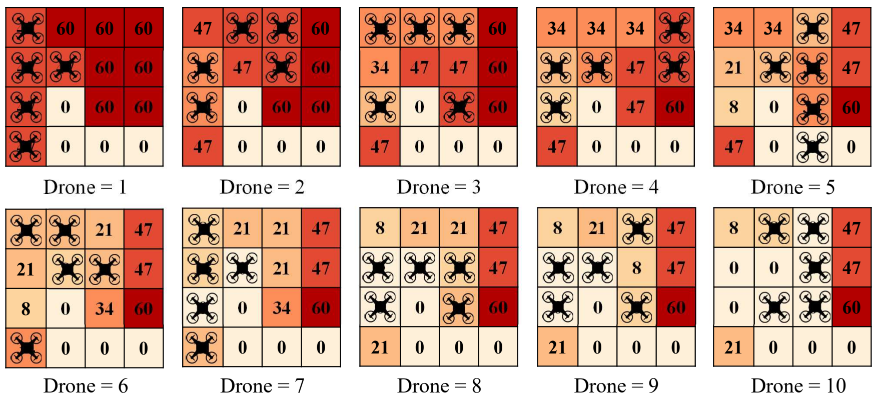
Table 6.1 shows the measurements made for the optimized navigation and tasking of each DJI Tello UAV: Battery consumption level, visited cells extracted from the selected plan, total traveling time (second), actual power consumption as well as the estimated hovering and maneuvering power (watt). The visited cells (see Fig. 6.5(b)) are extracted from the selected plans (non-zero values). The actual power consumption of the mission is influenced by the start and end battery level as voltage varies with the LiPo battery capacity[1] [194]. The hovering and maneuvering power consumption are calculated based on the earlier power consumption model [34] and that is why it remains constant across the drones with the same specification.

Fig. 6.8(a) illustrates a series of sensing maps depicting how drones meet the target sensing requirements after each trip by minimizing the mismatch between the total actual collected data and the required ones. This strategy ($\beta = 0$) is the one implemented in the indoor lab environment. To clearly show the effect of optimized sensing, a greedy strategy ($\beta = 1$) is shown in Fig. 6.8(b) in which drones select the plan with the low-

[1]http://en.fullymax.com/



(a) Coordinated optimized sensing with EPOS that minimizes sensing mismatch.

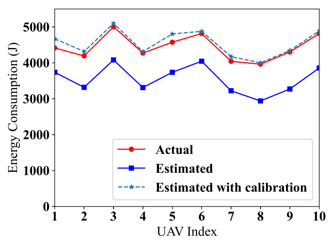


(b) Greedy sensing strategy that minimizes energy consumption.

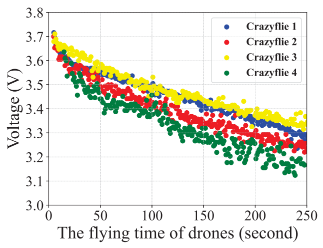
Figure 6.8: Coordinated vs. greedy sensing by a swarm of drones.

est energy consumption without any coordination. With coordination, the total energy consumption is $35.53kJ$ with a mismatch (RMSE) of 0.0057 and a mission inefficiency of 2.22%. The latter is defined by the ratio of required values in all cells that are not sensed by the drones during their mission over the total required values in all cells. In contrast, without coordination, the greedy strategy lowers energy consumption to $27.61kJ$ at a cost of higher mismatch (0.265) and mission inefficiency (26.11%). As can be observed in the series of sensing maps, coordination results in routing paths that expand further away from the departure/landing cell, while avoiding over-sensing/under-sensing that is likely to happen without coordination.

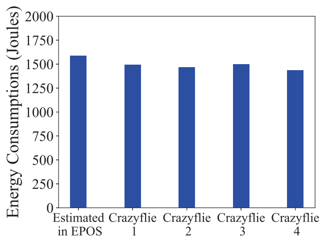
Fig. 6.9(a) compares, for each drone, the actual energy consumption with the model-



(a) Comparison of energy consumption of DJI Edu Tello.



(b) Actual voltage logging of Crazyflies.



(c) Real vs. Estimated energy consumption of Crazyflies.

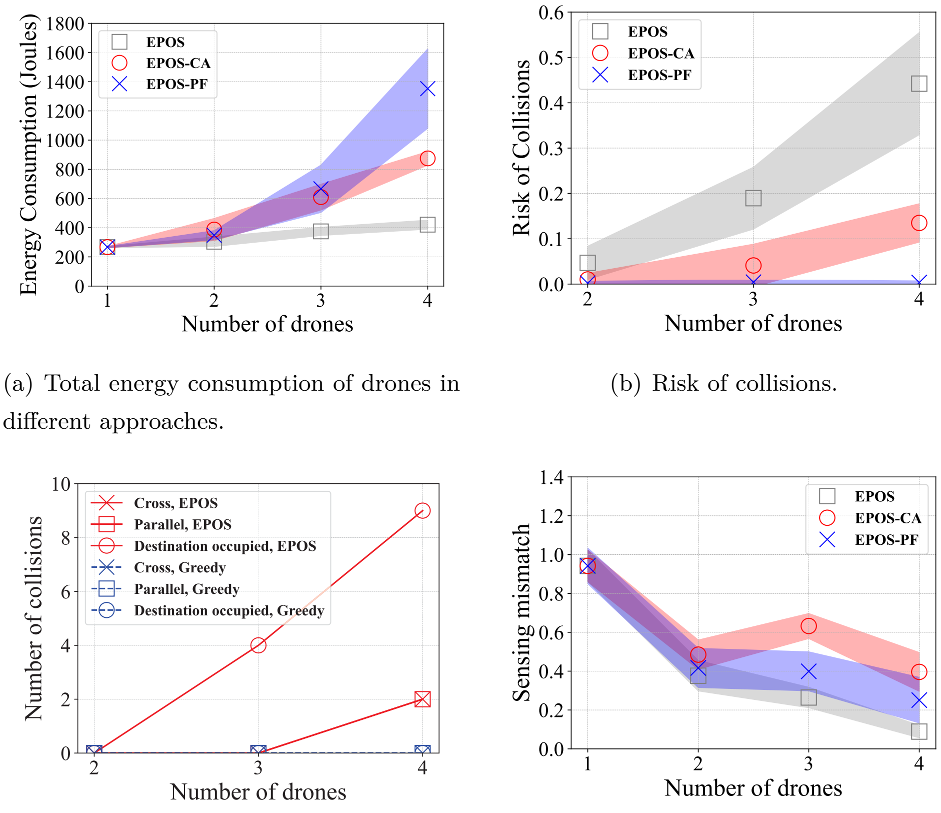
Figure 6.9: Comparison between estimated and real energy consumption.

based estimated one calculated during the planning phase. The actual energy consumption is initially found higher than the estimated one. This is because the drones spend some additional flying time to calibrate between departure and landing. To account for this additional energy consumption, the calibration time is recorded and added up to the original estimated energy consumption, resulting is a highly accurate estimation for all drones. A low error range from 37.41 to 255.52 Joule is attributed to the variable power consumption of the battery.

6.3.3 Evaluation on collision avoidance

To validate the applicability and realism of M-TET, the energy consumption of drones estimated in software is compared with the actual energy consumption in hardware. Fig. 6.9(b) illustrates the actual voltage of Crazyflies within the logging system of Crazyswarm where 4 Crazyflies only hover for 250 seconds. The voltage is recorded per second and is used to calculate the actual energy consumption based on battery capacity (250mAh) and expected flight time (7min). Fig. 6.9(c) compares, for each drone, the actual energy consumption with model-based estimated the one calculated during the planning phase of M-TET, with only approximately 120.4 joules error. This is because Crazyflies spend some additional flying time to calibrate between departure and landing.

Different methods are compared with 4 number of drones in total energy consumption. Firstly, the software runs with 40 different areas of the city centre [4], and then choose one of the results (the average of all results) for the execution in hardware. Fig. 6.10(a) shows that the disparity of energy consumption with and without collision



(a) Total energy consumption of drones in different approaches.

(b) Risk of collisions.

(c) Count of different types of collisions.

(d) Sensing mismatch.

Figure 6.10: Energy, sensing and collision performance comparison of drones.

avoidance rises as the number of drones increases, especially for artificial potential field. This is because each drone needs to detect and avoid several collisions within a small testbed, extending their traveling time. Noted that the points markers shown in figures denote hardware results whereas the shadow represents the errors in software results.

Albeit a high energy consumption, *EPOS-PF* significantly reduces the risk of collisions compared to other baseline methods, as shown in Fig. 6.10(b). It can detect various types of collisions that *EPOS-CA* cannot (e.g., the case where three drones collide). The low risk of collisions in *EPOS-PF* further proves the applicability of M-TET. Besides, since the highest number of collisions belongs to destination-occupied collisions, as shown in Fig. 6.10(c), drones with low priority in *EPOS-CA* continuously wait and sense the same cell, resulting in over-sensing and under-sensing. This increases the sensing mismatch of *EPOS-CA* compared to *EPOS* as shown in Fig. 6.10(d). In contrast, *EPOS-PF* dynamically repels and attracts drones to different areas, thereby mitigating over-sensing and under-sensing, and resulting in a sensing mismatch approximately 21.93% lower than *EPOS-CA*. *Greedy-PF* chooses the navigation and tasking with only one cell to minimize the energy consumption, mitigating the risk of collision, but still confronts over-sensing and under-sensing.

6.4 Comparison with Related Work

To validate realism and external validity of the algorithms to solve drone-based task allocation [3], previous work has built testbed for real-world implementation and assessment. Khan et al. [195] tests the feasibility of mobile target tracking algorithms using clustering and cover-set coverage methods with help of Parrot AR.Drone quadcopters. The testbed in [196] demonstrates the performance of a dedicated Quality-of-Service communication system using Paparazzi drones for cooperative sensing missions. However, these testbeds fail to emulate collisions with fixed or dynamic obstacles, leading to inaccuracies in analysis of multi-drone operations in real-world scenarios.

Collision avoidance algorithms find applicability to task allocation with drones in order to facilitate safe and reliable airspace access [197]. Computer vision and machine learning-based techniques that utilize a combination of sensors such as LiDAR, Radar and Sonar propose onboard collision avoidance strategies [197, 198]. Artificial potential fields is another widely known collision avoidance strategy, commonly used in robotics [36]. Batinovic et al. [152] uses artificial potential field for obstacle avoidance in 3D Environments by equipping drones with LiDAR sensors, and eliminated the possibility of agents getting stuck in a local minima employing rotational components of the repulsive force. Nevertheless, they only conduct experiments in simulation, lacking the demonstration in applicability and realism. Sabikan et al. [24] addressed this issue by developing a Time-to-Collision mathematical model using particle swarm optimization to find collision-free paths for outdoor drone data recording, but their work is limited to a platform with a single quadcopter. Schmittle et al. [25] establishes an open-source, cloud-enabled testbed to study the navigation around obstacles and drone swarm formation. However, it is limited to an exclusive software-based simulation and does not analyze how collision avoidance impacts the task allocation to drones, particularly regarding energy consumption and sensor data collection.

The designed testbed overcome these barriers by integrating potential field algorithm for collision avoidance with our testbed [26, 151]. This enhanced testbed can adapt to various application scenarios and contexts while ensuring safety during navigation and tasking. By enhancing realism, it effectively emulate the outdoor environments. Furthermore, it examines the impact of collision avoidance on energy consumption and data collection, offering new insights into hardware testing for drones.

6.5 Discussion and Future Work

In summary, **several new insights** on experimental results are: (1) M-TET using *EPOS-PF* improves task accuracy of traffic monitoring while avoiding collisions, making it energy-efficient with a limited number of drones. (2) The accurate estimation of energy consumption and low risk of collisions validate M-TET as a proof-of-concept, proving its feasibility and safety in real-word applications. (3) The expenditure of M-TET (each Crazyflie with necessary decks costs around $600) is significantly lower than outdoor experimentation with larger drones with cameras (e.g., Phantom 4 Pro at around $1600). (4) M-TET eliminates concerns about licensing, atmospheric conditions, and privacy violations.

The testbed opens up several promising avenues for further improvements. They are illustrated as follows: (1) Using advanced hardware (e.g., ultrasonic sensor) for real-time collision avoidance. (2) Expanding the experimentation in the delivery scenario using multiple drones. (3) Incorporating different drones with different sensors and data collection capabilities. (4) Implementing and integrating HALOP for enhanced online testbed operations in the evolving multi-agent systems.

6.6 Conclusions

In conclusion, this chapter introduces a novel testbed (M-TET) to integrate collision avoidance method to task allocation with drones. Designed for indoor lab environments, M-TET simplifies complex outdoor task scenarios while enhancing the realism of experimentation. As a proof-of-concept, this paper demonstrates the applicability of a collective learning algorithm and an artificial potential field algorithm to coordinate drones navigation and tasking for traffic monitoring. The results highlight the potential of M-TET and provide valuable opportunities for the broader community to enhance drone control in low-cost, safe and efficient task allocation scenarios for Smart Cities.

Chapter 7

Conclusions and Future Work

This thesis studies distributed multi-drone coordination for large-scale task allocation. It spans the full pipeline from generic model development and complex simulations to real-world adaptation under physical and operational constraints across diverse applications. The primary objective is to allocate tasks to swarms of drones, determining which points of interest to visit and when, to ensure energy-efficient and accurate mission completion. This involves the constraints such as limited battery, data collection requirements, delivery time window, payload capacities, recharging availability, collision avoidance in dynamic environments.

This chapter concludes the contributions and scientific findings of this thesis, and is organized as follows: Section 7.1 illustrates how the proposed approach of the thesis meets research objectives defined in Chapter 1. Section 7.2 presents theoretical, practical and ethical implications for broader research community and transportation applications. Finally, Section 7.3 discusses open issues and future work.

7.1 Meeting Research Objectives

Recall from Section 1.2 the main research question of this thesis:

Overarching Research Question: *How to design and prototype a multi-drone coordination system that supports scalable and adaptive task allocation across diverse applications?*

This thesis attempts to answer this question by meeting three sub-objectives defined in Section 1.2, which requires the distributed multi-drone coordination system: (1) To scale effectively in large spatio-temporal environments while ensuring long-term benefits, and satisfying hard constraints. (2) To adapt to diverse and dynamic real-world conditions such as energy constraints. (3) To become low costly, low risky, and easy to manage during hardware test, reducing fragmentation of simulation and real-world applications.

To meet the first objective, Chapter 3 introduces a novel, generic and highly-efficient distributed coordination model, named the Planning-based Multi-Agent Coordination (PMAC). This model employs a hierarchical framework with two high-level strategies (*grouping plan constraints* and *grouping behavior ranges*) to combine reinforcement learning (*MAPPO*) with optimized plan selection (*OPS*). PMAC retains the long-term foresight of *MAPPO*, allowing agents to anticipate future environmental changes and maximize long-term rewards. Meanwhile, it reduces computational/communication overhead by decreasing the joint action-state space, and preserves agent autonomy and privacy through *EPOS*. Experimental results on a synthetic dataset show that this hybrid approach (*HALOP*) significantly improves the system-wide performance compared to standalone *MAPPO* and *OPS* by around 31.29% and 23.69% respectively. PMAC also remains effective and scalable even when extended to hundreds of agents. Moreover, the incorporation of a hard constraint satisfaction mechanism boosts the system-wide constraint satisfaction rate by approximately 65%, demonstrating its robustness in constrained multi-agent environments.

To achieve the second objective, this thesis propose two application models extend the PMAC model to realistic scenarios: (i) the energy-aware coordination of multi-drone navigation and sensing model introduced in Chapter 4, and (ii) the energy-and-delay-aware coordination of drone multi-parcel delivery model introduced in Chapter 5. Both models incorporate a power consumption model [34], allowing each drone to estimate the energy consumption of a trip based on its physical characteristics (e.g., weight, propeller), environmental conditions (e.g., wind speed), and task-specific demands (e.g., payload weight). The long-term learning capability of PMAC further assists drones to predict evolving task requirements, such as changing traffic flow and customer requests, such that they can prioritize time-sensitive tasks and dynamically determine their flight range and recharging locations. As a result, the proposed approach significantly improves performance: in traffic monitoring, it achieves 23.17% and 27.83% higher overall performance in vehicle observation than *MAPPO* and *OPS* respectively. In the last-mile delivery scenario, it reduces delivery delays by approximately 0.44 hour over these baselines. Other experiments, including variations in the number of drones and base stations or depots, demonstrate the scalability and adaptability of PMAC across different real-world applications.

Lastly, to satisfy the third objective, Chapter 6 presents the Multi-drone Tasking Experimentation Testbed (M-TET), a first working prototype that validates the applicability and realism of PMAC. In the indoor lab setup, M-TET uses DJI and Crazyflie drones that are cheap, tiny and easily programmable, using optimized plan selection and artificial potential fields for task allocation and collision avoidance. As a proof-of-concept, the testbed successfully demonstrates accurate energy estimation (with a low error ranging from 0.74% to 5.1%) and low risk of navigation (nearly 100% of preventing collisions). This testbed validates that the conceptual and algorithmic contributions of PMAC are not limited to simulations but can operate effectively in controlled physical environments. M-TET showcases the feasibility of collective decision-making, energy-aware planning, and safe navigation among real drones, highlighting the practicality of deploying PMAC in real-world intelligent transportation systems.

7.1.1 Significance of the work

This thesis makes a significant contribution at the intersection of fundamental computer science and applied engineering, particularly in the fields of multi-agent systems, reinforcement learning, distributed optimization, and intelligent transportation. The core research has led to 4 high-quality publications [22, 26, 27, 40], including one that received a Best Paper Award at a leading international conference, as well as a registered patent, collectively highlighting the novelty and impact of the proposed approaches across both theory and application.

A comprehensive portfolio of experimental, empirical, and analytical evidence underscores the validity of this work. Results are drawn from extensive simulations and emulated environments (e.g., the developed indoor drone testbed), as well as from real-world datasets (e.g., large-scale traffic flow data from a European city). This experimental methodology ensures both robustness and realism in evaluating the performance of the system.

Moreover, the contributions go beyond theoretical insights to include practical implementations in both software and hardware. The work has produced 6 open-source code and datasets, with documentation to encourage further research and adoption.

A fully operational drone testbed and several demonstration videos have also been developed to showcase real-world feasibility (see List of Code/Data/Videos).

The broader impact of this work is further demonstrated through academic outreach and knowledge exchange. Research visits to institutions such as the University of Sheffield, LUT, and UPB fostered international collaboration [23, 26]. The findings have been successfully transferred into education by co-supervising a MEng Group Project at the University of Leeds, leading to an award-winning outcome [27]. The research has also been supported by multiple scholarships and funding schemes, including the EPS International PhD Studentship, the Alan Turing Mobility Scheme, the Enfield Exchange Scheme, and international travel grants, reflecting strong academic recognition and support.

7.2 Implications for Theory, Practice and Ethics

This thesis offers significant implications for policymakers, system operators, and technology designers by advancing theoretical understandings, practical deployment and ethical considerations of multi-drone coordination systems in transportation applications.

7.2.1 Theoretical implications

At the theoretical level, this work advances the field of multi-objective combinatorial optimization in evolving multi-agent systems. It is achieved by demonstrating the effectiveness of hierarchical coordination, where learning-based methods guide high-level decision-making, while static optimization techniques handle fine-grained task allocation. This synthetic solution proves more robust and efficient than either long-term learning or short-term optimization methods used in isolation. It allows high-level policies to dynamically adapt to changing environmental constraints, such as selecting navigation regions in a large urban area, while retaining the strengths of low-level optimization methods. For instance, the integration of collective learning contributes scalability (supporting a large number of agents), efficiency (through low computational/communication cost), and decentralization (preserving agent autonomy). Meanwhile, exact algorithms (e.g., greedy) ensure the satisfaction of hard or critical constraints that are otherwise difficult to guarantee using learning methods alone.

The system design introduced in this thesis contributes a methodologically rigorous approach to drone-based task allocation problem. It starts by classifying the tasks based on their dynamics and constraint types, and abstracting them into a unified mathematical model. Suitable state-of-the-art algorithms are then selected or extended to solve these problem classes effectively. The approach is then grounded in real-world application domains, where additional scenario-specific constraints are modeled to increase realism. This application-informed algorithmic design cycle, where practical requirements shape theoretical development and are in turn validated in realistic scenarios, offers a valuable design approach for researchers. It bridges the gap between abstraction and deployment, guiding future work in high-level decision-making for drone-based applications and other domains involving autonomous intelligent systems.

7.2.2 Practical implications

Practically, the scenarios studied in this thesis meet net zero targets in intelligent transportation systems [112]. For instance, the proposed system enables drones to efficiently monitor traffic conditions, allowing for early detection of congestion or accidents. Proper mitigation actions taken by traffic operators can reduce carbon emissions. In the last-mile delivery, the system fulfills delivery requests in a timely and energy-efficient manner, which decreases the carbon footprint by alleviating the workload of heavy ground vehicles, such as vans and trucks.

The demonstrated scalability and adaptability to complex environments of the system, such as varying drone densities, diverse task requirements, and the presence of multiple recharging stations and depots, support its robustness in real-world deployments. The analysis of drone density shows that learning-based approaches perform better under scarce drone resources, helping drones learn environmental patterns and prioritize critical tasks. This reduces the need for large fleets, lowering capital expenditure on drones, particularly when executing large spatio-temporal missions. Particularly, the proposed method exhibits good applicability in urgent medical deliveries in scenarios such as pandemic. It reduces the expected delivery delay of customers, leading to higher customer satisfaction and retention, ultimately enhancing long-term quality of service. Moreover, the methods with ability to direct drones toward task-dense or high-priority areas offer insights for infrastructure planning, such as optimal placement of recharging stations or depots.

From an implementation perspective, the development of the indoor multi-drone experimentation testbed (M-TET) provides a low-cost, safe, and privacy-preserving alternative to outdoor experiments. It avoids environmental constraints including airspace regulation, unpredictable weather and public surveillance concerns, while enabling realistic evaluation of drone coordination approaches. By using affordable hardware (e.g., Crazyflie and DJI drones) in a controlled lab setting, this testbed lowers the entry barrier for research and industrial experimentation, offering a replicable and scalable prototype for broader adoption.

7.2.3 Ethical implications

As drone technologies rapidly evolve, from basic remote operations to more controversial uses such as surveillance and military, it is the responsibility to consider the humanitarian and ethical dimensions of autonomous drone deployment [199]. In particular, the proposed distributed multi-drone coordination system is designed with strong consideration for civilian privacy, safety, and equality.

To protect citizens' privacy, drones in the system are restricted to fly to the residential areas by setting the hard constraints. This not only prevent from giving citizens the impression of surveillance, but controls drones to operate at low disturbance times. Moreover, drones hover at around 164.8m when performing traffic monitoring (see Chapter 4), adhere to regulations such as the guidelines of UK Civil Aviation Authority: drones must remain "over or within 150m of any congested area" or "within 50 m of any person", avoiding direct interference with public spaces [200, 201].

Importantly, the drones employed in this system are small and explicitly designed for civilian applications (e.g., traffic monitoring, last-mile delivery) in controlled, safe environments. They are not intended for military use [202, 203]. For example, logistics drones are limited to carrying parcels and performing infrastructure inspections. The open distributed approach also significantly reduces the risk of centralized misuse or coordinated attacks by malicious actors [202, 203].

In addition, the proposed approach is fully open-source and supported by a low-cost, easily replicable testbed prototype, encouraging researchers and practitioners to adopt and extend the system with minimal barriers. By making the platform accessible to all, this thesis promotes inclusive innovation and helps prevent power imbalances in the development and application of drone technologies.

7.3 Open Issues and Future work

While this thesis proposes a scalable and adaptive multi-drone coordination system, several open issues remain that warrant further research, particularly in the areas of algorithmic design and real-world applicability.

The PMAC model offers a structured approach to multi-drone task allocation. However, it currently relies on centralized training for high-level planning strategies ($HALOP$), which introduces potential vulnerabilities to adversarial attacks and limits the resilience of the model. To mitigate this, future work should explore fully decentralized training paradigms [204], where agents learn and update strategies without centralized control. A major challenge here is designing efficient and scalable information exchange protocols among agents that preserve learning performance without overloading communication networks. Similarly, the plan selection should reduce the reliance on the centralized exact algorithms. This requires an improved hard constraint satisfaction approach for decentralized combinatorial optimization to address critical constraints. Moreover, decentralized coordination still faces security and trust challenges during inter-agent communication. Ensuring that drones share only essential and trustworthy information without exposing sensitive data is critical. Emerging techniques such as blockchain-based consensus, federated learning, or zero-knowledge proofs could help establish secure and verifiable cooperation among drones, promoting both robustness and privacy [205–208].

Although the proposed PMAC model offers promising coordination strategies for drone-based transportation applications such as urban sensing and last-mile delivery, several practical aspects remain underexplored. Notably, the current task allocation models assume ideal environmental conditions and do not account for real-world constraints such as obstacles (e.g., buildings), dynamic wind conditions, or no-fly zones. While the M-TET testbed introduces collision avoidance using artificial potential fields, this operates after the task allocation stage and may lead to plan violations. To address this, future research should focus on integrating environmental constraints directly into the task planning and selection process. Drones should be equipped with adaptive planning capabilities that allow them to dynamically adjust flight paths, such as altering altitude or rerouting, to avoid collisions and maintain mission objectives. Such integration will ensure the generated plans remain executable and safe in real-world conditions.

The experimental evaluations in this thesis offer valuable insights but remain limited in scope. Future work should expand both the breadth and realism of experimentation. This includes: (1) Combine multiple types of sensors (e.g., cameras, LiDAR, thermal) to enable multi-modal sensing for more comprehensive urban data collection, such as temperature and humidity. Note that it is not time-consuming to develop a new drone-based model using PMAC. The interfaces of the OPS or HALOP approach are generalized while only involving a new dataset for experimental evaluation. (2) Coordinate heterogeneous drones (e.g., different sizes, speeds, and capabilities) for delivery tasks with varied parcel sizes, delivery times, and energy constraints. (3) Extend the testbed from offline to online by considering the drone-based task offloading of edge-to-edge and edge-to-cloud. For example, each edge is customized to a drone by running a software agent for communication, plan generation and selection, while a cloud could store replay buffer for MARL training. (4) Using Large Language Model (LLM) to tune heuristics (collective learning or exact algorithms) for users based on a specific problem in real scenario and explain why a drone swarm made certain choices. LLM can also convert high-level human instructions (e.g., survey the busiest roads and deliver parcels to hospital first) into machine-readable task allocations for drones.

In conclusion, advancing from intelligent coordination algorithms to real-world drone flight not only bridges the gap between theory and practice, but also paves the way for social impact, enabling safer, greener, and more efficient urban transportation services through autonomous aerial systems.

Appendix A

Supplementary Material of Chapter 3

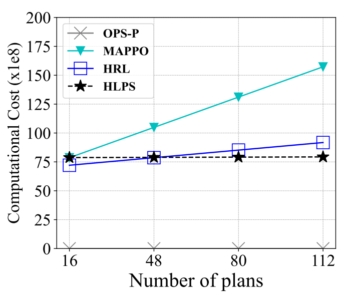
A.1 Additional Experimental Evaluation

This appendix evaluates the performance of the proposed *HALOP* in the following aspects: 1) computation and communication overhead, 2) performance crossover point, and 3) training convergence.

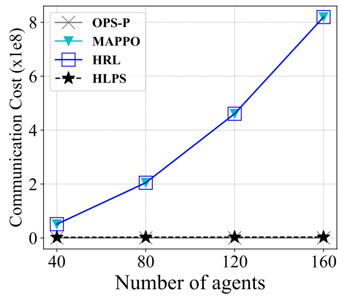
Computation and communication overhead. Fig. A.1(a) illustrates that *HALOP* significantly reduces computation cost during training compared to *MAPPO* and *HRL*. As the number of plans increases, *MAPPO* exhibits a near-linear growth in computation, reaching nearly twice the cost of *HALOP* at 112 plans, highlighting the effectiveness of the proposed strategy for action space reduction. Fig. A.1(b) shows that *HALOP* achieves lower communication overhead than *MAPPO*, which scales exponentially with the number of agents. This underscores the advantage of decentralized coordination in communication efficiency.

Training convergence. Fig. A.1(c) shows the training process of four MARL-based methods. With the help of low-level multi-agent collective learning, *HALOP* achieves faster exploration and converges around 600 episode, significantly earlier than *MAPPO*, which converges after 2000 episode. Note that *HRL* converges the slowest due to its non-stationary learning process.

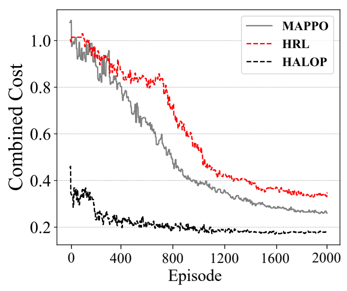
Performance crossover point. Fig. A.2(a), A.2(b) and A.2(c) compare the cost performance of different methods over time. The proposed *HALOP* shows consistent improvement and eventually surpasses *OPS-P*. Specifically, by time period 4, *HALOP* achieves a lower mean discomfort cost than *OPS-P*, and although its inefficiency cost exceeds that of *OPS-P* at time period 13, its discomfort cost remains 66.31% lower, resulting in a lower combined cost. These findings highlight the long-term benefits of high-level strategy learning in multi-agent collective learning, particularly over extended time horizons.



(a) Computation overhead vs. number of plans.



(b) Communication overhead vs. number of agents.



(c) Training process.

Figure A.1: Computation overhead comparison of all methods.

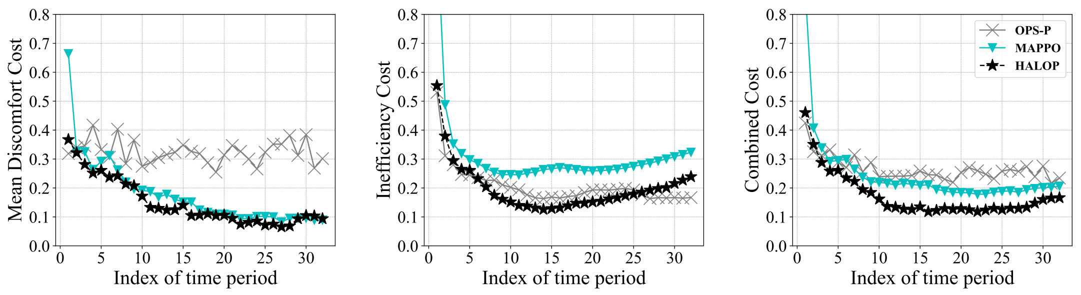


Figure A.2: The first three figures are the cost performance comparison of methods per time period.

Furthermore, the impact of various parameters within the proposed *HALOP* is analyzed. The insights derived from this appendix can be used to make informed empirical decisions regarding parameter selection for the evaluation scenarios, including the number of actions $|A|$, agent behavior,β and the weights σ_1, σ_2 in reward function. The calculations presented herein can be automated for any scenario during hyper-parameter optimization. Such optimization is not the focus of this paper. All experiments are performed in the basic synthetic scenario (40 agents, 16 plans per agent and the target signal with $\omega = \pi/24$).

Agent behavior. Fig. A.3 illustrates the effect of agent behavior by varying the parameter β_t^u defined in Eq.(4.24). As β_t^u increases from 0 to 1, agents behave selfishly, and thus reduce the mean discomfort cost while enhancing the inefficiency cost. To minimize the combined cost, $\beta_t^u = 0.5$ is selected in *OPS* and *HALOP-P* as it serves as Pareto optimal point.

Weights in reward function. Fig. A.4 shows the effect of weight parameters by varying the σ_1 defined in Eq.(3.15), and $\sigma_2 = 1 - \sigma_1$. As σ_1 increases from 0 to 1, the weight on discomfort cost in the reward function increases, thereby reducing mean discomfort cost while increasing inefficiency cost in *MAPPO* and *HALOP*. This paper chooses $\sigma_1 = \sigma_2 = 0.5$ since *HALOP* achieves the minimum combined cost (Pareto efficiency).

Number of actions. Fig. A.5 shows the cost comparison of the number of actions of the proposed methods. In *HALOP-P*, when the number of plan groups increases, agents can accurately find the plans with lower discomfort cost, as shown in Fig.A.5(b), but maintain inefficiency cost. The average frequency denotes the total number of actions taken by all agents over all time periods divided by the number of sub-ranges covered by each group/behavior range. In *HALOP-B*, when the number of behavior ranges increases, agents can sample from a narrow range of behaviors and precisely lock on an optimal one (see Fig.A.5(d)). However, with more plan groups and behavior ranges, the costs of these methods increase since high action space makes the training inefficient (see the convergence of *MAPPO* in Fig. A.1(c)).

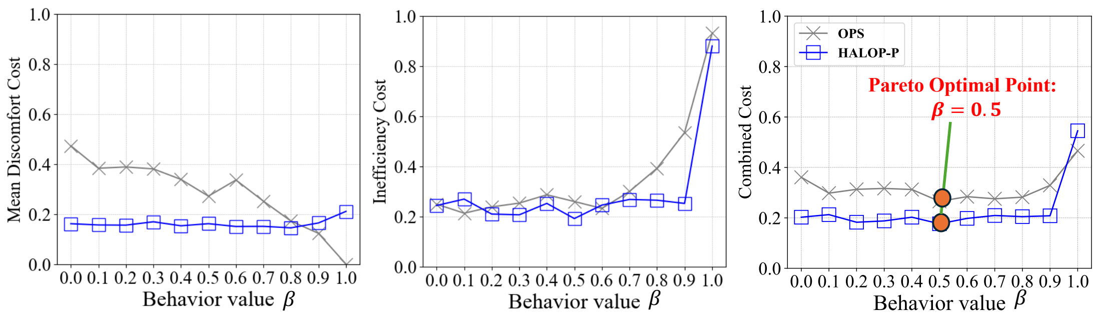


Figure A.3: The Pareto optimality of *OPS* and *HALOP-P*.

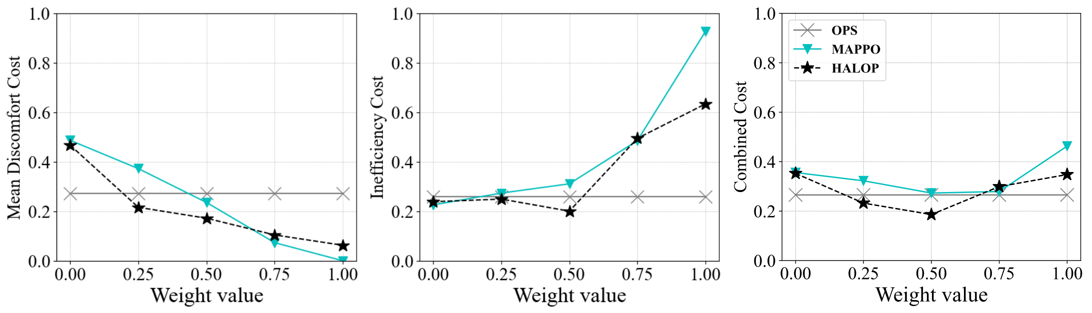
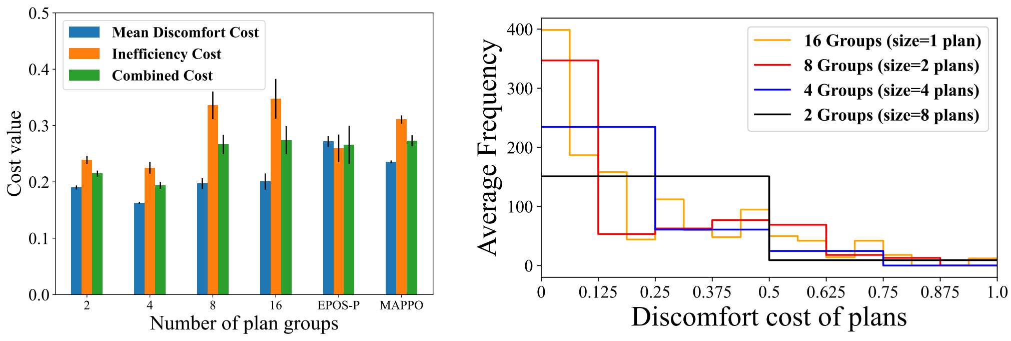
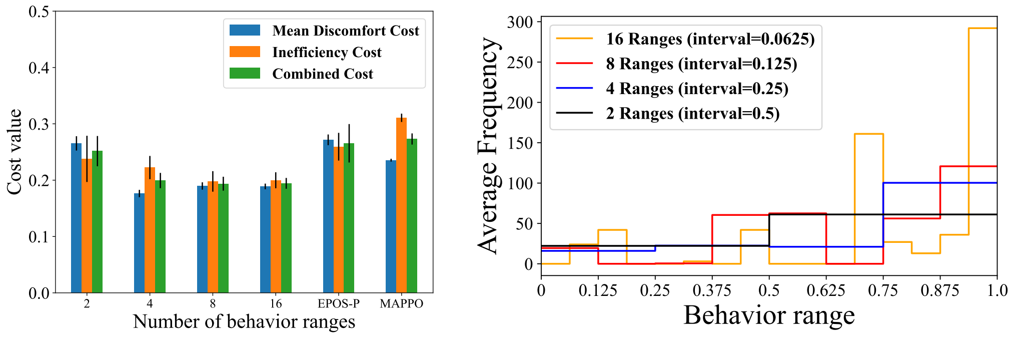


Figure A.4: Cost comparison of *OPS*, *MAPPO* and *HALOP* as the increase of weight.



(a) Number of plan groups. (b) Average frequency vs. discomfort cost of groups.



(c) Number of behavior ranges. (d) Average frequency vs. behavior range.

Figure A.5: Costs and average frequency of *HALOP-P* with different number of plan groups and *HALOP-B* with different number of behavior ranges.

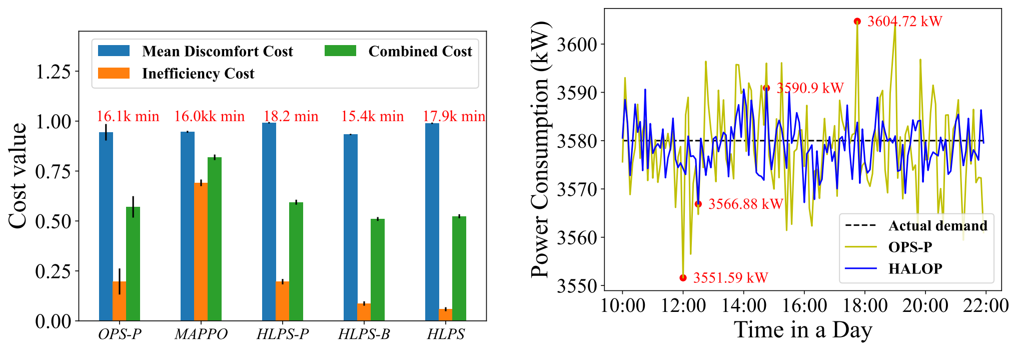
A.2 Evaluation with Smart Grids

The PCMF framework is applicable in the broader context of large-scale and complex multi-agent systems. This appendix shows the broad and significant impact of the proposed generic algorithm in the context of smart grids and smart cities, i.e., energy management.

The energy application scenario uses a dataset derived by energy disaggregation of the simulated zonal power transmission in the Pacific Northwest Smart Grid Demonstrations Project [65, 136, 209]. It contains consumers (agents) who participate equipped with one or more controllable household appliances, e.g., refrigerators, water heaters, heating/cooling systems, and so on [65]. The energy demand (kW) of consumers, which is recorded every $5min$ in a $12h$ span of a day, is modeled as a plan. The plans of each consumer represent different energy consumption patterns generated using a load-shifting strategy for managing electricity use across peak and off-peak times, prompting for grid load balance [65]. Specifically, the measured demand is the first plan; the other 9 plans are generated by shifting the positions of values in the first plan, i.e., shifting the measured demand 75, 150 or 720 minutes. The discomfort cost of each plan is the amount of minutes shifted compared to the original demand.

In this scenario, each consumer selects a plan of energy demand for $12h$ (i.e., a time period from 10:00 to 22:00 in a day) and aggregate the plans selected by others. The total energy demand is obtained by adding up the selected plans element-wise per time period. The goal of system is to mitigate power peaks and reduce the risk of blackouts [18] by minimizing the variance of the total energy demand at each time. For example, if consumers notice that energy demand is higher in the morning before 12:00 compared to other times of the day, they may learn to select a plan with lower energy demand during the morning hours on the following day. Meanwhile, consumers aim to secure their energy needs on peak times and avoid disrupted (or shifted) plans, often prioritizing comfort at the expense of system-wide efficiency and balance. Thus, their objective is to decrease their amount of minutes shifted by choosing a plan with lower discomfort cost.

Fig. A.6(a) shows that the proposed methods outperform *MAPPO* in terms of inefficiency cost, achieving approximately 36.03% lower combined cost, particularly due to the efficient coordination through multi-agent collective learning. Several other metrics are measured, including (a) the mean minutes shifted per agent, and (b) max and



(a) The cost comparison.

(b) Total energy demands at each time.

Figure A.6: Performance comparison of all methods in the energy management scenario (160 consumers, 10 plans per consumer and 16 time periods, i.e., 16 days).

min power peak over all periods. Among all methods, *HALOP-B* achieves the lowest discomfort and inefficient costs. Compared to *HALOP-P*, *HALOP-B* helps agents to stay closer to their original energy demands, reducing total deviation time by over 2.8k minutes across all time periods, significantly enhancing consumer satisfaction. Furthermore, compared to *OPS-P*, *HALOP-B* reduces max power peaks by 13.82kW and enhances min power peaks by 15.29kW, see Fig. A.6(b), thereby improving the power grid stability. In this scenario, the *grouping behavior ranges* strategy outperforms *grouping plans*. These findings suggest that, in environments with a limited number of plans per agent and where Pareto efficiency is the primary goal, it is more effective to group behavior ranges rather than plans.

Appendix B

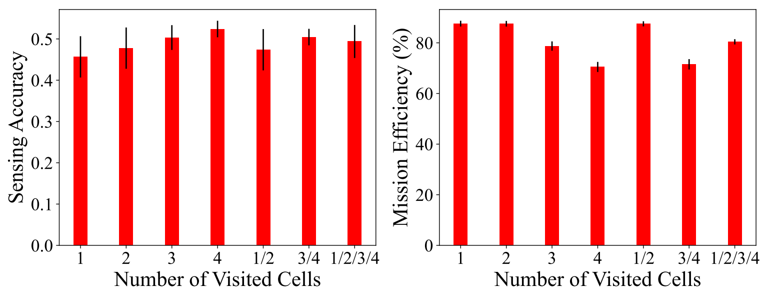
Supplementary Material of Chapter 4

B.1 Effect of different parameters

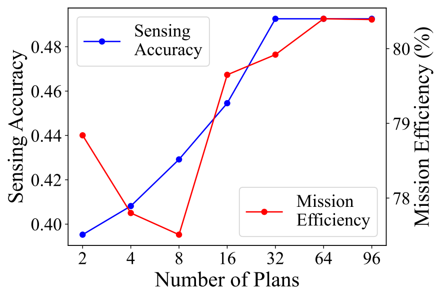
The purpose of the evaluation in this section is to understand how different parameters influence the system performance. The results of this section are used to make an empirical choice of these parameters for the rest of the evaluation scenarios. The calculations presented here can be automated for any scenario within a hyper-parameter optimization, which is though not the focus of this paper.

Mobility range in short-term. Three policies of plan generation are compared in EPOS: *balance*, *min sensing mismatch* and *min sensing inefficiency*. A simple test is implemented in the basic synthetic scenario to compare different numbers of visited cells set in the input, see Algorithm 1. As shown in Fig. B.1(a), the higher the number of cells a drone visits, the higher the flying energy is, and the lower the allocated hovering/sensing energy is, which increases the mission inefficiency. To balance the sensing mismatch and mission inefficiency, the policies of min sensing mismatch ($|J_u| = 1|2$, the strategy chooses 1 or 2 cells randomly), min mission inefficiency ($|J_u| = 3|4$), and balance ($|J_u| = 1|2|3|4$) are empirically designed. Thus, three policy-based methods for EPOS are proposed that are referred to as *EPOS-mismatch*, *EPOS-inefficiency* and *EPOS-balance* respectively.

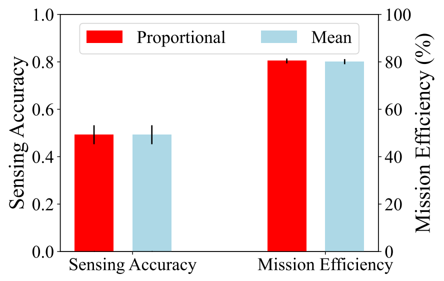
Mobility range in long-term. Fig. B.2(a) illustrates the effect of different numbers of visited cells. The higher the number of cells a drone visits, the higher the flying energy is, and the higher the energy cost is. The high number of visited cells perplexes the spatio-temporal navigation and sensing of a drone, leading to over-sensing. Specifically, multiple drones visit the same cell simultaneously and waste energy on collecting the



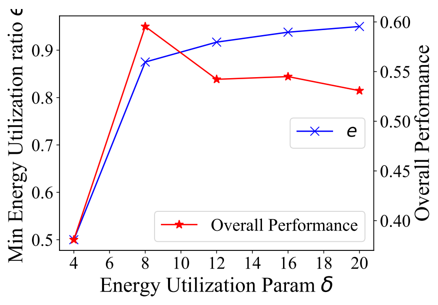
(a) Mobility range.



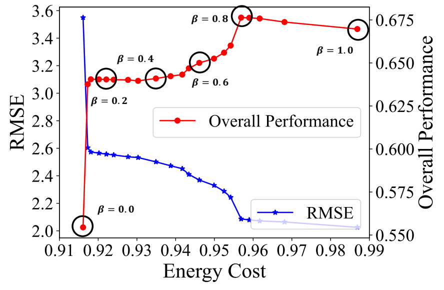
(b) Number of plans.



(c) Sensing allocation schemes.



(d) Energy Utilization.



(e) Agents' behavior (β).

Figure B.1: Performance comparison for different parameters of the proposed method.

same data. In contrast, if drones visit only one cell, they are free from over-sensing, but have a high probability to miss the area with a high required sensing data due to its low mobility range. As a result, $J(a^u) = 2$ is empirically selected for each drone to optimize the overall performance.

Number of plans. See Fig. B.1(b), as the number of generated plans increases, both sensing mismatch and mission inefficiency decrease and converge to 64. Thus, 64 plans for each agent are generated in EPOS.

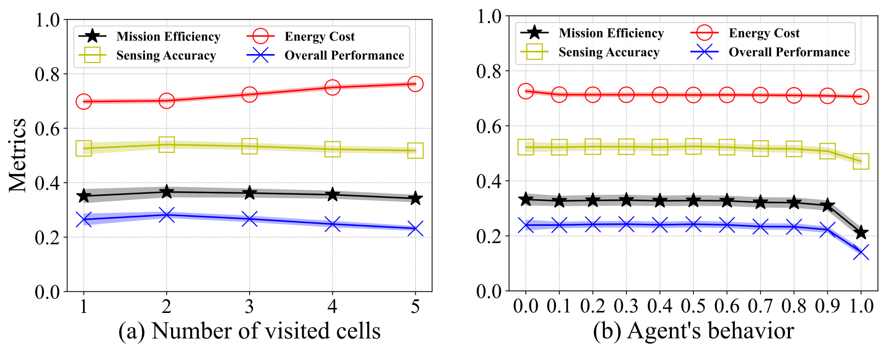


Figure B.2: Change the parameters of *DO-RL* in mobility range and agents' behavior.

Sensing allocation: proportional vs. mean. Fig. B.1(c) illustrates the performance comparison between proportional and mean sensing allocation defined in Eq.(4.18). The mean sensing allocation has higher inefficiency than proportional one. Using the Mann-Whitney U test, the sensing mismatch distribution of the two methods comes with $p = 0.08$, the one of mission inefficiency has $p = 0.003$. The results demonstrate a statistically higher performance when sensor values collected are proportional, i.e. a similar mission mismatch but a 1.56% higher mission efficiency.

Energy utilization. Fig. B.1(d) illustrates the changes of minimum energy utilization ratio e (when $p = P$ in Eq.(4.15)) and the overall performance as the parameter δ increases. According to the results, a $\delta = 8$ is selected for the experimental settings.

Agents' behavior in short-term. Fig. B.1(e) illustrates the effect of agents' behavior by varying the parameter β [18]. As β increases from 0 to 1, agents reduce the energy cost of their selected plans, while increasing the sensing mismatch. This is because drones with higher β choose a plan with lower energy cost, i.e., the plan with lower energy utilization ratio e, and the total sensing values collected is reduced according to Eq. (4.16) and (4.17). To minimize the log_{10} RSS, the value of $\beta = 0$ is selected in the proposed methods, while $\beta = 0.2$ is the calculated Pareto optimal point referred to as *EPOS-Pareto*, minimizing the combined cost.

Agents' behavior in long-term. Fig. B.2(b) illustrates the effect of agents' behavior by varying the parameter of β [18]. As β increases from 0 to 1, agents reduce the energy cost of their selected plans, while decreasing both efficiency and accuracy. This is because, according to Eq. 4.24, drones with higher β choose a plan with lower energy cost, which degrades the matching between the total sensing and the target. Since the overall performance for $\beta \in [0.1, 0.8]$ is statistically similar, using the Mann-Whitney U test with average p-value of 0.0003, drones are allowed to choose their behaviors within this range.

Hyperparameter in deep reinforcement learning. Table B.1 shows the training performance of different hyperparameter in DRL, including the discount factor γ, batch size H, clip interval ϵ, and deep neutral networks. The proposed approach takes the parameters of $\gamma = 0.95$, $H = 64$, $\epsilon = 0.2$, and RNN neutral network. It compares the reward and the episode to converge by changing these parameters. The results demonstrate a more stable (lower reward error and higher converged episode) but lower average reward in training when decreasing γ, ϵ or increasing H. In contrast, a relatively higher average reward leads to higher error and slower convergence. In addition, the RNN used in the proposed *HALOP* performs better than the multilayer perceptron (MLP), with 7.8% higher average reward. Considering these results, the parameters of $\gamma = 0.95$, $H = 64$, $\epsilon = 0.2$, and RNN neutral network are selected for *HALOP*.

Table B.1: Reward and convergence vs. hyperparameter.

Approaches **Attributes.:**	Proposed	$\gamma = 0.90$	$\gamma = 0.99$	$H = 32$	$H = 128$	$\epsilon = 0.1$	$\epsilon = 0.3$	MLP
Average reward	8.42	8.45	8.24	8.30	8.42	8.17	8.34	7.81
Error of reward	0.68	0.79	0.66	0.66	0.71	0.68	0.87	0.55
Converged episode	1899	2048	2451	1071	2589	1843	2116	1657

B.2 Mobility and Sensing Quality

Two theorems are introduced that link the performance metrics of mission inefficiency and sensing mismatch with the mobility of the coordinated drones, in particular with their flying coverage modelled by the number of visiting cells.

Theorem B.2.1. *In a mission of a swarm of drones $\mathcal{U}$, starting from their base stations, flying with a constant ground speed over an area consisting of N cells to collect sensor data, and returning back consuming all energy of their battery, the mission in-*

efficiency is proportional to the number of random visited cells $|J_u|$*:*

$$\alpha_1 \cdot |J_u| \to f_1(J_u) = 1 - \frac{\sum_{u\in\mathcal{U}} V(J_u)}{\sum_{n\in\mathcal{N}} R_n}, \tag{B.1}$$

if the drones collect sensor values over the visited cells proportionally to required target values, where α_1 *is a positive constant and* $f_1(J_u)$ *is the function of mission inefficiency according to the optimization objective of Eq.(4.6).*

Proof. Based on Eq. (4.16) and (4.17) and since each drone uses all its battery capacity for flying and hovering, the number of collected sensor values over the cells J_u is:

$$V(J_u) = \sum_{n\in\mathcal{N}} S_{u,n} = \frac{C_u \cdot e - P_u^{\mathrm{f}} \cdot \tau(J_u)}{P_u^{\mathrm{h}}} \cdot f. \tag{B.2}$$

Eq.(4.6) can be reformulated as:

$$1 - \frac{\sum_{u\in\mathcal{U}} \sum_{n\in\mathcal{N}} V_n^u}{\sum_{n\in\mathcal{N}} R_n} = 1 - \frac{\sum_{u\in\mathcal{U}} V(J_u)}{\sum_{n\in\mathcal{N}} R_n} := f_1(J_u), \tag{B.3}$$

given that $\sum_{u\in\mathcal{U}} \sum_{n\in\mathcal{N}} V_n^u = \sum_{u\in\mathcal{U}} V(J_u)$. The total flying time $\tau(J_u)$ of a drone u can be modelled as:

$$\tau(J_u) \approx (|J_u| - 1) \cdot \tilde{\tau} + 2 \cdot \tilde{\tau}_B, \tag{B.4}$$

where $\tilde{\tau}$ is the mean expected traveling time between any two random cells and $\tilde{\tau}_B$ is the mean expected traveling time between the base station of the drone and a random cell. Assume the number of random visited cells increases from $|J_u|$ to $|J'_u|$, where $|J_u| < |J'_u|$. Then, the total flying time without hovering is also likely to increase as $\tau(J_u) < \tau(J'_u)$ given that each drone u flies with a constant ground speed. The influence $f_1(J'_u) - f_1(J_u)$ on the mission inefficiency based on Eq.(B.3) is calculated as follows:

$$\begin{aligned} f_1(J'_u) - f_1(J_u) &= \frac{\sum_{u\in\mathcal{U}}(\overbrace{V(J_u)}^{Eq.(B.2)} - \overbrace{V(J'_u)}^{Eq.(B.2)})}{\sum_{n\in\mathcal{N}} R_n} \\ &= \frac{\sum_{u\in\mathcal{U}} \frac{P_u^{\mathrm{f}} \cdot f}{P_u^{\mathrm{h}}}}{\sum_{n\in\mathcal{N}} R_n} \cdot (\overbrace{\tau(J'_u)}^{Eq.(B.4)} - \overbrace{\tau(J_u)}^{Eq.(B.4)}) \\ &\approx \frac{\sum_{u\in\mathcal{U}} \frac{P_u^{\mathrm{f}} \cdot f}{P_u^{\mathrm{h}}}}{\sum_{n\in\mathcal{N}} R_n} \cdot \tilde{\tau} \cdot (|J'_u| - |J_u|) \\ &\leftarrow \alpha_1 \cdot (|J'_u| - |J_u|). \end{aligned} \tag{B.5}$$

Since all parameters in $\frac{\sum_{u\in\mathcal{U}} \frac{P_u^{\mathrm{f}} \cdot f}{P_u^{\mathrm{h}}}}{\sum_{n\in\mathcal{N}} R_n} \cdot \tilde{\tau}$ are positive, $\alpha_1 > 0$, and therefore the mission inefficiency is proportional to the number of random visited cells. □

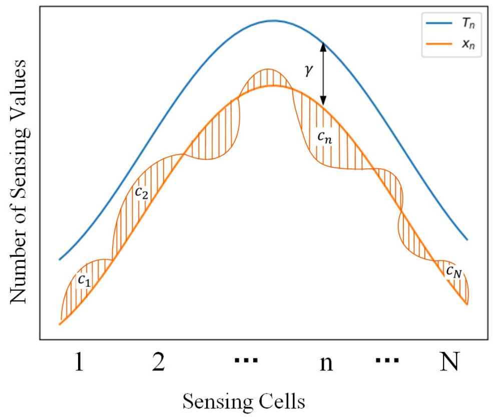


Figure B.3: The distribution of targets and total sensing values collected by drones.

Theorem B.2.2. *In a mission of a swarm of drones $\mathcal{U}$, starting from their base stations, flying with a constant ground speed over an area consisting of N cells to collect sensor data, and returning back consuming all energy of their battery, the sensing mismatch is inverse proportional to the number of random visited cells $|J_u|$:*

$$\alpha_2 \cdot |J_u| \to f_2(J_u) = \sum_{n \in \mathcal{N}} (R_n - \sum_{u \in \mathcal{U}} V_n^u)^2, \tag{B.6}$$

if and only if $|J'_u| + |J_u| < N$ when increasing the number of visited cells from $|J_u|$ to $|J'_u|$ and if the drones collect sensor values over the visited cells proportionally to required target values, where α_2 is a negative constant and $f_2(J_u)$ is a function of sensing mismatch according to the optimization objective of Eq. (4.7).

Proof. Figure B.3 assists this proof. Since sensor values over the different cells are collected proportionally to the required target values R_n, the collected sensor values are modelled by $x_n + v_n = \sum_{u \in \mathcal{U}} V_n^u$, $\forall n \in \{1, ..., N\}$, and $x_n \leq R_n$. Each x_n corresponds to the collected sensor values with an optimal matching to the required target values (min RSS), while $v_n \in \{+c_n, -c_n, 0\}$ models mismatches such that $\sum_{n \in \mathcal{N}} (x_n + v_n) \approx \sum_{n \in \mathcal{N}} x_n$, and thus $\sum_{n \in \mathcal{N}} v_n \approx 0$. Moreover, the mission inefficiency in the optimal collected sensor data x_n is given by γ such that $\gamma := R_n - x_n$. The optimal matching between R_n and x_n denotes that $\gamma \geq 0$ is constant $\forall n \in \{1, ..., N\}$. Therefore, it holds that:

$$\begin{aligned}\sum_{n\in\mathcal{N}}(R_n - \sum_{u\in\mathcal{U}} V_n^u) &= \sum_{n\in\mathcal{N}}(R_n - (x_n + v_n)) \\ &\approx \sum_{n\in\mathcal{N}}(R_n - x_n) \\ &\approx N\cdot\gamma > 0\end{aligned} \tag{B.7}$$

Eq.(B.7) can be squared to calculate the sensing mismatch $f_2(J_u)$ as follows:

$$\begin{aligned}f_2(J_u) &= \sum_{n\in\mathcal{N}}(R_n - (x_n + v_n))^2 \\ &= \sum_{n\in\mathcal{N}}(\gamma - v_n)^2 \\ &= N\cdot\gamma^2 - 2\gamma\cdot\sum_{n\in\mathcal{N}} v_n + \sum_{n\in\mathcal{N}} v_n^2 \\ &= N\cdot\gamma^2 + \sum_{n\in\mathcal{N}} c_n^2.\end{aligned} \tag{B.8}$$

The higher the c_n is, the higher the $f_2(J_u)$.

The distribution of the mission inefficiency values $R_n - (x_n + v_n)$ among N cells is determined by the selection of the cells by each drone. By assuming that each of the U drones chooses the visiting cells randomly (with replacement), the distribution among N cells can be explained by a Binomial distribution:

$$P(X = k, J_u) = \binom{U}{k}\cdot p(J_u)^k\cdot(1 - p(J_u))^{U-k}, \tag{B.9}$$

where $p(J_u)$ is the probability that a drone u chooses $|J_u|$ number of cells from the total of N cells that do not contain the cell n. This results in mismatch at cell n that either originates from (i) an under-sensing $v_n = -c_n$, if a drone u has a high probability $p(J_u)$ to miss cell n from J_u, or (ii) an over-sensing $R_n - (x_n + v_n) = C + c_n$ if this probability is low (see Fig. B.3). The probability $p(J_u)$ can be formulated as:

$$p(J_u) = \binom{N-1}{|J_u|}\Big/\binom{N}{|J_u|} = 1 - \frac{|J_u|}{N}, \tag{B.10}$$

which expresses that the higher the number of visiting cells is, the lower the probability of drone u to miss a cell n.

The mismatch c_n at cell n for a drone visiting J_u points can be modeled by a Binomial distribution:

$$c_n(J_u) := k_n(J_u)\cdot\overline{S}_n(J_u), \tag{B.11}$$

with the expected values of $k_n(J_u)$ denoting the average number of drones that miss cell n and $\overline{S}_n(J_u)$ denoting the average number of collected values by each drone u. The expressions of these values are formulated as follows:

$$k_n(J_u) = U \cdot p(J_u), \tag{B.12}$$

$$\overline{S}_n(J_u) = \sum_{u\in\mathcal{U}} V(J_u) \cdot \frac{R_n}{U \cdot \sum_{n\in\mathcal{N}} R_n}. \tag{B.13}$$

By increasing the number of sensing cells from $|J_u|$ to $|J'_u|$, the influence $f_2(J'_u) - f_2(J_u)$ on the sensing mismatch can be formulated using Eq. (B.8) as follows:

$$\begin{aligned}
f_2(J'_u) - f_2(J_u) &= \sum_{n\in\mathcal{N}} [\overbrace{c_n(J'_u)^2}^{Eq.(B.11,B.12,B.13)} - \overbrace{c_n(J_u)^2}^{Eq.(B.11,B.12,B.13)}] \\
&= \sum_{n\in\mathcal{N}} \frac{R_n}{\sum_{n\in\mathcal{N}} R_n} \cdot \overbrace{p(J'_u)^2}^{Eq.(B.10)} \cdot \overbrace{\sum_{u\in\mathcal{U}} V(J'_u)^2}^{Eq.(B.5)} \\
&\quad - \sum_{n\in\mathcal{N}} \frac{R_n}{\sum_{n\in\mathcal{N}} R_n} \cdot \overbrace{p(J_u)^2}^{Eq.(B.10)} \cdot \overbrace{\sum_{u\in\mathcal{U}} V(J_u)^2}^{Eq.(B.5)} \\
&\leftarrow \sum_{n\in\mathcal{N}} R_n \cdot [\frac{\alpha_1}{N} \cdot |J'_u|^2 - \alpha_1 \cdot |J'_u|] - \sum_{n\in\mathcal{N}} R_n \cdot [\frac{\alpha_1}{N} \cdot |J_u|^2 - \alpha_1 \cdot |J_u|] \\
&\leftarrow [\frac{|J'_u| + |J_u|}{N} - 1] \cdot \alpha_1 \cdot \sum_{n\in\mathcal{N}} R_n \cdot (|J'_u| - |J_u|) \\
&\leftarrow \alpha_2 \cdot (|J'_u| - |J_u|)
\end{aligned} \tag{B.14}$$

Thus, it holds that $\alpha_2 = [\frac{|J'_u|+|J_u|}{N} - 1] \cdot \alpha_1 \cdot \sum_{n\in\mathcal{N}} R_n < 0$ if and only if $|J'_u| + |J_u| < N$, where in this case the sensing mismatch is proven to be reverse proportional to the number of random visited cells J_u. □

References

[1] Hamid Menouar, Ismail Guvenc, Kemal Akkaya, A Selcuk Uluagac, Abdullah Kadri, and Adem Tuncer. UAV-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28, 2017.

[2] Fei-Yue Wang, Yilun Lin, Petros A Ioannou, Ljubo Vlacic, Xiaoming Liu, Azim Eskandarian, Yisheng Lv, Xiaoxiang Na, David Cebon, Jiaqi Ma, et al. Transportation 5.0: The dao to safe, secure, and sustainable intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 24(10): 10262–10278, 2023.

[3] Yongkun Zhou, Bin Rao, and Wei Wang. UAV swarm intelligence: Recent advances and future trends. *IEEE Access*, 8:183856–183878, 2020.

[4] Emmanouil Barmpounakis and Nikolas Geroliminis. On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment. *Transportation research part C: emerging technologies*, 111:50–71, 2020.

[5] Sabitri Poudel and Sangman Moh. Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey. *Vehicular Communications*, page 100469, 2022.

[6] Zhi Pei, Tao Fang, Kebiao Weng, and Wenchao Yi. Urban on-demand delivery via autonomous aerial mobility: Formulation and exact algorithm. *IEEE Transactions on Automation Science and Engineering*, 20(3):1675–1689, 2022.

[7] Jun Tang, Haibin Duan, and Songyang Lao. Swarm intelligence algorithms for

multiple unmanned aerial vehicles collaboration: A comprehensive review. *Artificial Intelligence Review*, 56(5):4295–4327, 2023.

[8] Yu Bai, Hui Zhao, Xin Zhang, Zheng Chang, Riku Jäntti, and Kun Yang. Toward autonomous multi-UAV wireless network: A survey of reinforcement learning-based approaches. *IEEE Communications Surveys & Tutorials*, 25(4):3038–3067, 2023.

[9] Yang Gao, Yingzhou Zhang, Shurong Zhu, and Yi Sun. Multi-UAV task allocation based on improved algorithm of multi-objective particle swarm optimization. In *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 443–4437. IEEE, 2018.

[10] Xueli Wu, Yanan Yin, Lei Xu, Xiaojing Wu, Fanhua Meng, and Ran Zhen. Multi-UAV task allocation based on improved genetic algorithm. *IEEE Access*, 9:100369–100379, 2021.

[11] Chengyi Qu, Rounak Singh, Alicia Esquivel Morel, Francesco Betti Sorbelli, Prasad Calyam, and Sajal K Das. Obstacle-aware and energy-efficient multi-drone coordination and networking for disaster response. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 446–454. IEEE, 2021.

[12] Sayani Sarkar, Michael W Totaro, and Khalid Elgazzar. Intelligent drone-based surveillance: application to parking lot monitoring and detection. In *Unmanned Systems Technology XXI*, volume 11021, pages 13–19. SPIE, 2019.

[13] Francesco Betti Sorbelli. Uav-based delivery systems: A systematic review, current trends, and research challenges. *Journal on Autonomous Transportation Systems*, 1(3):1–40, 2024.

[14] Shumaila Javaid, Nasir Saeed, Zakria Qadir, Hamza Fahim, Bin He, Houbing Song, and Muhammad Bilal. Communication and control in collaborative UAVs: Recent advances and future trends. *IEEE Transactions on Intelligent Transportation Systems*, 24(6):5719–5739, 2023.

[15] Paolo Toth and Daniele Vigo. An overview of vehicle routing problems. *The vehicle routing problem*, pages 1–26, 2002.

[16] Chuhao Qin and Evangelos Pournaras. Strategic coordination for evolving multi-agent systems: A hierarchical reinforcement and collective learning approach. *arXiv preprint arXiv:2509.18088*, 2025.

[17] Evangelos Pournaras. Multi-level reconfigurable self-organization in overlay services. 2013. Ph.D. Dissertation. TU Delft, Delft University of Technology.

[18] Evangelos Pournaras, Peter Pilgerstorfer, and Thomas Asikis. Decentralized collective learning for self-managed sharing economies. *ACM Transactions on Autonomous and Adaptive Systems*, 13(2):1–33, 2018.

[19] Yunhao Yang and Andrew Whinston. A survey on reinforcement learning for combinatorial optimization. In *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pages 131–136. IEEE, 2023.

[20] Roberto Roberti and Mario Ruthmair. Exact methods for the traveling salesman problem with drone. *Transportation Science*, 55(2):315–335, 2021.

[21] Norzailawati Mohd Noor, Alias Abdullah, and Mazlan Hashim. Remote sensing uav/drones and its applications for urban areas: A review. In *IOP conference series: Earth and environmental science*, volume 169, page 012003. IOP Publishing, 2018.

[22] Chuhao Qin and Evangelos Pournaras. Coordination of drones at scale: Decentralized energy-aware swarm intelligence for spatio-temporal sensing. *Transportation Research Part C: Emerging Technologies*, 157:104387, 2023.

[23] Chuhao Qin, Arun Narayanan, and Evangelos Pournaras. Coordinated multi-drone last-mile delivery: Learning strategies for energy-aware and timely operations. *arXiv preprint arXiv:2509.15830*, 2025.

[24] Sulaiman Bin Sabikan, Sophan Wahyudi Nawawi, and NAA Aziz. Modelling of time-to collision for unmanned aerial vehicle using particles swarm optimization. *IAES International Journal of Artificial Intelligence*, 9(3):488, 2020.

[25] Matt Schmittle, Anna Lukina, Lukas Vacek, Jnaneshwar Das, Christopher P Buskirk, Stephen Rees, Janos Sztipanovits, Radu Grosu, and Vijay Kumar. OpenUAV: A UAV testbed for the CPS and robotics community. In *2018 ACM/IEEE*

9th International Conference on Cyber-Physical Systems (ICCPS), pages 130–139. IEEE, 2018.

[26] Chuhao Qin, Fethi Candan, Lyudmila Mihaylova, and Evangelos Pournaras. 3, 2, 1, Drones go! A testbed to take off UAV swarm intelligence for distributed sensing. In *UK Workshop on Computational Intelligence*, pages 576–587. Springer, 2022.

[27] Chuhao Qin, Alexander Robins, Callum Lillywhite-Roake, Adam Pearce, Hritik Mehta, Scott James, Tsz Ho Wong, and Evangelos Pournaras. M-SET: Multi-drone swarm intelligence experimentation with collision avoidance realism. In *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pages 1–7. IEEE, 2024.

[28] Yongbo Chen, Di Yang, and Jianqiao Yu. Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, 54 (6):2853–2872, 2018.

[29] Jie Chen, Xianguo Qing, Fang Ye, Kai Xiao, Kai You, and Qian Sun. Consensus-based bundle algorithm with local replanning for heterogeneous multi-UAV system in the time-sensitive and dynamic environment. *The Journal of Supercomputing*, 78(2):1712–1740, 2022.

[30] Guohua Wu, Mingfeng Fan, Jianmai Shi, and Yanghe Feng. Reinforcement learning based truck-and-drone coordinated delivery. *IEEE Transactions on Artificial Intelligence*, 4(4):754–763, 2021.

[31] John W Creswell. *Educational research: Planning, conducting, and evaluating quantitative and qualitative research.* pearson, 2015.

[32] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, pages 75–105, 2004.

[33] Chuhao Qin and Evangelos Pournaras. Short vs. long-term coordination of drones: When distributed optimization meets deep reinforcement learning. *arXiv preprint arXiv:2311.09852*, 2023.

[34] Joshuah K Stolaroff, Constantine Samaras, Emma R O'Neill, Alia Lubers, Alexandra S Mitchell, and Daniel Ceperley. Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nature Communications*, 9 (1):1–13, 2018.

[35] Chuhao Qin and Evangelos Pournaras. EPOS-based Plans for Drones. 12 2022. doi: 10.6084/m9.figshare.21432804.v17. URL https://figshare.com/articles/dataset/EPOS-based_Plans_for_Drones/21432804.

[36] Giuseppe Fedele, Luigi D'Alfonso, Francesco Chiaravalloti, and Gaetano D'Aquila. Obstacles avoidance based on switching potential functions. *Journal of Intelligent & Robotic Systems*, 90:387–405, 2018.

[37] Junayed Pasha, Zeinab Elmi, Sumit Purkayastha, Amir M Fathollahi-Fard, Ying-En Ge, Yui-Yip Lau, and Maxim A Dulebenets. The drone scheduling problem: A systematic state-of-the-art review. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14224–14247, 2022.

[38] Shahad Alqefari and Mohamed El Bachir Menai. Multi-UAV task assignment in dynamic environments: Current trends and future directions. *Drones*, 9(1):75, 2025.

[39] Evangelos Pournaras. Collective learning: A 10-year odyssey to human-centered distributed intelligence. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 205–214. IEEE, 2020.

[40] Srijoni Majumdar, Chuhao Qin, and Evangelos Pournaras. Discrete-choice multi-agent optimization: Decentralized hard constraint satisfaction for smart cities. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 60–76. Springer, 2023.

[41] Hoai An Le Thi, Duc Manh Nguyen, and Tao Pham Dinh. Globally solving a nonlinear UAV task assignment problem by stochastic and deterministic optimization approaches. *Optimization Letters*, 6:315–329, 2012.

[42] Sahar Kouroshnezhad, Ali Peiravi, Mohammad Sayad Haghighi, and Alireza Jolfaei. An energy-aware drone trajectory planning scheme for terrestrial sensors localization. *Computer Communications*, 154:542–550, 2020.

[43] Omar Bouhamed, Xiangpeng Wan, Hakim Ghazzai, and Yehia Massoud. A DDPG-based approach for energy-aware UAV navigation in obstacle-constrained environment. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.

[44] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV coverage path planning under varying power constraints using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1444–1449. IEEE, 2020.

[45] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV path planning using global and local map information with deep reinforcement learning. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 539–546. IEEE, 2021.

[46] Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pages 438–445. IEEE, 2004.

[47] Byung Duk Song, Jonghoe Kim, Jeongwoon Kim, Hyorin Park, James R Morrison, and David Hyunchul Shim. Persistent UAV service: An improved scheduling formulation and prototypes of system components. *Journal of Intelligent & Robotic Systems*, 74(1):221–232, 2014.

[48] Mehdi Alighanbari and Jonathan How. Robust decentralized task assignment for cooperative UAVs. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6454, 2006.

[49] Qiao Cheng, Dong Yin, Jian Yang, and Lincheng Shen. An auction-based multiple constraints task allocation algorithm for multi-uav system. In *2016 International Conference on Cybernetics, Robotics and Control (CRC)*, pages 1–5. IEEE, 2016.

[50] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Distributed algorithms for multirobot task assignment with task deadline constraints. *IEEE Transactions on Automation Science and Engineering*, 12(3):876–888, 2015.

[51] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.

[52] Xiaowei Fu, Peng Feng, and Xiaoguang Gao. Swarm UAVs task and resource dynamic assignment algorithm based on task sequence mechanism. *IEEE Access*, 7:41090–41100, 2019.

[53] Wenfei Wang, Maolong Lv, Le Ru, Bo Lu, Shiguang Hu, and Xinlong Chang. Multi-UAV unbalanced targets coordinated dynamic task allocation in phases. *Aerospace*, 9(9):491, 2022.

[54] Wanqing Zhao, Qinggang Meng, and Paul WH Chung. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. *IEEE transactions on cybernetics*, 46(4):902–915, 2015.

[55] Rahim Ali Qamar, Mubashar Sarfraz, Atta Rahman, and Sajjad A Ghauri. Multi-criterion multi-UAV task allocation under dynamic conditions. *Journal of King Saud University-Computer and Information Sciences*, 35(9):101734, 2023.

[56] Ruchir Patel, Eliot Rudnick-Cohen, Shapour Azarm, Michael Otte, Huan Xu, and Jeffrey W Herrmann. Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3770–3776. IEEE, 2020.

[57] S Burak Akat and Veysel Gazi. Decentralized asynchronous particle swarm optimization. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8. IEEE, 2008.

[58] Arushi Gupta and Smriti Srivastava. Comparative analysis of ant colony and particle swarm optimization algorithms for distance optimization. *Procedia Computer Science*, 173:245–253, 2020.

[59] Weinan Wu, Jie Xu, and Yiming Sun. Integrate assignment of multiple heterogeneous unmanned aerial vehicles performing dynamic disaster inspection and validation task with dubins path. *IEEE Transactions on Aerospace and Electronic Systems*, 59(4):4018–4032, 2023.

[60] Lei Cao, He shun Tan, Hui Peng, and Ming cong Pan. Multiple UAVs hierarchical dynamic task allocation based on PSO-FSA and decentralized auction. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 2368–2373. IEEE, 2014.

[61] Jun Tang, Xi Chen, Xiaomin Zhu, and Feng Zhu. Dynamic reallocation model of multiple unmanned aerial vehicle tasks in emergent adjustment scenarios. *IEEE Transactions on Aerospace and Electronic Systems*, 59(2):1139–1155, 2022.

[62] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, 7(3–4):197–387, 2014.

[63] Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. Cohda: A combinatorial optimization heuristic for distributed agents. In *International Conference on Agents and Artificial Intelligence*, pages 23–39. Springer, 2013.

[64] Evangelos Pournaras, Srivatsan Yadhunathan, and Ada Diaconescu. Holarchic structures for decentralized deep learning: A performance analysis. *Cluster Computing*, 23(1):219–240, 2020.

[65] Evangelos Pournaras, Mark Yao, and Dirk Helbing. Self-regulating supply–demand systems. *Future Generation Computer Systems*, 76:73–91, 2017.

[66] Astrid Nieße, Michael Sonnenschein, Christian Hinrichs, and Jörg Bremer. Local soft constraints in distributed energy scheduling. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1517–1525. IEEE, 2016.

[67] Christian Hinrichs et al. A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents. *International Journal of Bio-Inspired Computation*, 10(2):69–78, 2017.

[68] Farzam Fanitabasi, Edward Gaere, and Evangelos Pournaras. A self-integration testbed for decentralized socio-technical systems. *Future Generation Computer Systems*, 113:541–555, 2020.

[69] Evangelos Pournaras, Mark Christopher Ballandies, Stefano Bennati, and Chien-fei Chen. Collective privacy recovery: Data-sharing coordination via decentralized artificial intelligence. *PNAS nexus*, 3(2):page 029, 2024.

[70] Evangelos Pournaras, Seoho Jung, Srivatsan Yadhunathan, Huiting Zhang, and Xingliang Fang. Socio-technical smart grid optimization via decentralized charge control of electric vehicles. *Applied soft computing*, 82:105573, 2019.

[71] Farzam Fanitabasi and Evangelos Pournaras. Appliance-level flexible scheduling for socio-technical smart grid optimization. *IEEE Access*, 8:119880–119898, 2020.

[72] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11):4948, 2021.

[73] Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 2024.

[74] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6):1291–1307, 2012.

[75] Haining Tan. Reinforcement learning with deep deterministic policy gradient. In *2021 International conference on artificial intelligence, big data and algorithms (CAIBDA)*, pages 82–85. IEEE, 2021.

[76] Zifan Wu, Chao Yu, Deheng Ye, Junge Zhang, Hankz Hankui Zhuo, et al. Coordinated proximal policy optimization. *Advances in Neural Information Processing Systems*, 34:26437–26448, 2021.

[77] Wei Dai, Huimin Lu, Junhao Xiao, Zhiwen Zeng, and Zhiqiang Zheng. Multi-robot dynamic task allocation for exploration and destruction. *Journal of Intelligent & Robotic Systems*, 98:455–479, 2020.

[78] Ziwei Liu, Changzhen Qiu, and Zhiyong Zhang. Sequence-to-sequence multi-agent reinforcement learning for multi-UAV task planning in 3D dynamic environment. *Applied Sciences*, 12(23):12181, 2022.

[79] Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):73–84, 2018.

[80] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.

[81] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[82] Federico Venturini, Federico Mason, Francesco Pase, Federico Chiariotti, Alberto Testolin, Andrea Zanella, and Michele Zorzi. Distributed reinforcement learning for flexible UAV swarm control with transfer learning capabilities. In *Proceedings of the 6th ACM workshop on micro aerial vehicle networks, systems, and applications*, pages 1–6, 2020.

[83] Dezhi Chen, Qi Qi, Zirui Zhuang, Jingyu Wang, Jianxin Liao, and Zhu Han. Mean field deep reinforcement learning for fair and efficient UAV control. *IEEE Internet of Things Journal*, 8(2):813–828, 2020.

[84] Jiaping Xiao, Phumrapee Pisutsin, and Mir Feroskhan. Collaborative target search with a visual drone swarm: An adaptive curriculum embedded multistage reinforcement learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[85] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

[86] Imen Jendoubi and François Bouffard. Multi-agent hierarchical reinforcement learning for energy management. *Applied Energy*, 332:120500, 2023.

[87] Zhiwei Xu, Yunpeng Bai, Bin Zhang, Dapeng Li, and Guoliang Fan. Haven: Hierarchical cooperative multi-agent reinforcement learning with dual coordination mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11735–11743, 2023.

[88] Caleb Chuck, Supawit Chockchowwat, and Scott Niekum. Hypothesis-driven skill discovery for hierarchical deep reinforcement learning. In *2020 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 5572–5579. IEEE, 2020.

[89] Baolai Wang, Shengang Li, Xianzhong Gao, and Tao Xie. UAV swarm confrontation using hierarchical multiagent reinforcement learning. *International Journal of Aerospace Engineering*, 2021(1):3360116, 2021.

[90] Zun Liu, Yuanqiang Cao, Jianyong Chen, and Jianqiang Li. A hierarchical reinforcement learning algorithm based on attention mechanism for UAV autonomous navigation. *IEEE Transactions on Intelligent Transportation Systems*, 24(11): 13309–13320, 2022.

[91] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2021.

[92] Marco Rinaldi, Sheng Wang, Renan Sanches Geronel, and Stefano Primatesta. Application of task allocation algorithms in multi-UAV intelligent transportation systems: A critical review. *Big Data and Cognitive Computing*, 8(12):177, 2024.

[93] Momena Monwar, Omid Semiari, and Walid Saad. Optimized path planning for inspection by unmanned aerial vehicles swarm with energy constraints. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[94] Jingyu He, Yao Xiao, Corina Bogdan, Shahin Nazarian, and Paul Bogdan. A design methodology for energy-aware processing in unmanned aerial vehicles. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27 (1):1–20, 2021.

[95] Sungjae Lee and Yosoon Choi. Comparison of topographic surveying results using a fixed-wing and a popular rotary-wing unmanned aerial vehicle (drone). *Tunnel and Underground Space*, 26(1):24–31, 2016.

[96] Di Wu, Dmitri I Arkhipov, Minyoung Kim, Carolyn L Talcott, Amelia C Regan, Julie A McCann, and Nalini Venkatasubramanian. ADDSEN: Adaptive data processing and dissemination for drone swarms in urban sensing. *IEEE Transactions On Computers*, 66(2):183–198, 2016.

[97] Yoshio Inoue. Satellite-and drone-based remote sensing of crops and soils for smart farming–a review. *Soil Science and Plant Nutrition*, 66(6):798–810, 2020.

[98] Eugen Valentin Butilă and Răzvan Gabriel Boboc. Urban traffic monitoring and analysis using unmanned aerial vehicles (UAVs): A systematic literature review. *Remote Sensing*, 14(3):620, 2022.

[99] Lige Ding, Dong Zhao, Mingzhe Cao, and Huadong Ma. When crowdsourcing meets unmanned vehicles: Toward cost-effective collaborative urban sensing via deep reinforcement learning. *IEEE Internet of Things Journal*, 8(15):12150–12162, 2021.

[100] Dong Zhao, Mingzhe Cao, Lige Ding, Qiaoyue Han, Yunhao Xing, and Huadong Ma. Dronesense: Leveraging drones for sustainable urban-scale sensing of open parking spaces. In *IEEE INFOCOM - Conference on Computer Communications*, pages 1769–1778. IEEE, 2022.

[101] Moataz Samir, Chadi Assi, Sanaa Sharafeddine, Dariush Ebrahimi, and Ali Ghrayeb. Age of information aware trajectory planning of UAVs in intelligent transportation systems: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 69(11):12382–12395, 2020.

[102] Zhenyu Zhou, Junhao Feng, Bo Gu, Bo Ai, Shahid Mumtaz, Jonathan Rodriguez, and Mohsen Guizani. When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning. *IEEE Transactions on Communications*, 66 (11):5526–5538, 2018.

[103] Igor Bisio, Chiara Garibotto, Halar Haleem, Fabio Lavagetto, and Andrea Sciarrone. A systematic review of drone based road traffic monitoring system. *IEEE Access*, 10:101537–101555, 2022.

[104] Fatma Outay, Hanan Abdullah Mengash, and Muhammad Adnan. Applications of unmanned aerial vehicle (UAV) in road safety, traffic and highway infrastructure management: Recent advances and challenges. *Transportation research part A: policy and practice*, 141:116–129, 2020.

[105] Allister Loder, Thomas Otte, and Klaus Bogenberger. Using large-scale drone

data to monitor and assess the behavior of freight vehicles on urban level. *Transportation Research Record*, 2676(11):496–507, 2022.

[106] Jasso Espadaler-Clapés, Emmanouil Barmpounakis, and Nikolas Geroliminis. Empirical investigation of lane usage, lane changing and lane choice phenomena in a multimodal urban arterial. *Transportation research part A: policy and practice*, 172:103674, 2023.

[107] Mouna Elloumi, Riadh Dhaou, Benoit Escrig, Hanen Idoudi, and Leila Azouz Saidane. Monitoring road traffic with a UAV-based system. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.

[108] Murat Bakirci. Enhancing vehicle detection in intelligent transportation systems via autonomous UAV platform and YOLOv8 integration. *Applied Soft Computing*, 164:112015, 2024.

[109] Hamed Jahani, Yunes Khosravi, Bahareh Kargar, Kok-Leong Ong, and Sobhan Arisian. Exploring the role of drones and uavs in logistics and supply chain management: A novel text-based literature review. *International Journal of Production Research*, 63(5):1873–1897, 2025.

[110] Samantha K Brooks, Rebecca K Webster, Louise E Smith, Lisa Woodland, Simon Wessely, Neil Greenberg, and Gideon James Rubin. The psychological impact of quarantine and how to reduce it: rapid review of the evidence. *The lancet*, 395 (10227):912–920, 2020.

[111] Zeashan Hameed Khan, Afifa Siddique, and Chang Won Lee. Robotics utilization for healthcare digitization in global COVID-19 management. *International journal of environmental research and public health*, 17(11):3819, 2020.

[112] Janfizza Bukhari, Abhishek G Somanagoudar, Luyang Hou, Omar Herrera, and Walter Mérida. Zero-emission delivery for logistics and transportation: Challenges, research issues, and opportunities. *The Palgrave Handbook of Global Sustainability*, pages 1729–1749, 2023.

[113] Kevin Dorling, Jordan Heinrichs, Geoffrey G Messier, and Sebastian Magierowski.

Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2016.

[114] Fangyu Hong, Guohua Wu, Qizhang Luo, Huan Liu, Xiaoping Fang, and Witold Pedrycz. Logistics in the sky: A two-phase optimization approach for the drone package pickup and delivery system. *IEEE Transactions on Intelligent Transportation Systems*, 24(9):9175–9190, 2023.

[115] Dyutimoy Nirupam Das, Rohan Sewani, Junwei Wang, and Manoj Kumar Tiwari. Synchronized truck and drone routing in package delivery logistics. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5772–5782, 2020.

[116] Shan-Huen Huang, Ying-Hua Huang, Carola A Blazquez, and Chia-Yi Chen. Solving the vehicle routing problem with drone for delivery services using an ant colony optimization algorithm. *Advanced Engineering Informatics*, 51:101536, 2022.

[117] Okan Dukkanci, Bahar Y Kara, and Tolga Bektaş. Minimizing energy and cost in range-limited drone deliveries with speed optimization. *Transportation Research Part C: Emerging Technologies*, 125:102985, 2021.

[118] Guillem Muñoz, Cristina Barrado, Ender Çetin, and Esther Salami. Deep reinforcement learning for drone delivery. *Drones*, 3(3):72, 2019.

[119] Farabi Ahmed Tarhan and Nazım Kemal Ure. Genetic-algorithm-aided deep reinforcement learning for multi-agent drone delivery. *Drones*, 8(3):71, 2024.

[120] Sung Hoon Chung, Bhawesh Sah, and Jinkun Lee. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, 123:105004, 2020.

[121] Júlia C Freitas, Puca Huachi V Penna, and Túlio AM Toffolo. Exact and heuristic approaches to truck–drone delivery problems. *EURO Journal on Transportation and Logistics*, 12:100094, 2023.

[122] Yaohan Shen, Bipan Zou, René De Koster, and TCE Cheng. Performance estimation and operating policies in a truck-based autonomous mobile robot delivery system. *International Journal of Production Research*, pages 1–23, 2024.

[123] Monica Gentili, Pitu B Mirchandani, Alessandro Agnetis, and Zabih Ghelichi. Locating platforms and scheduling a fleet of drones for emergency delivery of perishable items. *Computers & Industrial Engineering*, 168:108057, 2022.

[124] Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. A decentralized heuristic for multiple-choice combinatorial optimization problems. In *Operations Research Proceedings 2012: Selected Papers of the International Annual Conference of the German Operations Research Society (GOR), Leibniz University of Hannover, Germany, September 5-7, 2012*, pages 297–302. Springer, 2013.

[125] Manh Duong Phung and Quang Phuc Ha. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Applied Soft Computing*, 107:107376, 2021.

[126] Anand Nayyar and Rajeshwar Singh. A comprehensive review of ant colony optimization (ACO) based energy-efficient routing protocols for wireless sensor networks. *International Journal of Wireless Networks and Broadband Technologies (IJWNBT)*, 3(3):33–55, 2014.

[127] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3243–3250, 2020.

[128] Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A Cire. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3677–3687, 2021.

[129] Sijia Xu, Hongyu Kuang, Zhuang Zhi, Renjie Hu, Yang Liu, and Huyang Sun. Macro action selection with deep reinforcement learning in starcraft. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pages 94–99, 2019.

[130] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[131] Maheed A Ahmed and Mahsa Ghasemi. Privacy-preserving decentralized actor-critic for cooperative multi-agent reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2755–2763. PMLR, 2024.

[132] Jovan Nikolic and Evangelos Pournaras. Structural self-adaptation for decentralized pervasive intelligence. In *22nd Euromicro Conference on Digital System Design (DSD)*, pages 562–571. IEEE, 2019.

[133] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[134] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[135] Bingqing Jiang, Jun Du, Chunxiao Jiang, Zhu Han, and Merouane Debbah. Underwater searching and multi-round data collection via AUV swarms: An energy-efficient AoI-aware MAPPO approach. *IEEE Internet of Things Journal*, 2023.

[136] Evangelos Pournaras. Agent-based Planning Portfolio. 4 2019. doi: 10.6084/m9.figshare.7806548.v6. URL `https://figshare.com/articles/dataset/Agent-based_Planning_Portfolio/7806548`. DOI: https://doi.org/10.6084/m9.figshare.7806548.v6.

[137] Wenjie Yi, Rong Qu, Licheng Jiao, and Ben Niu. Automated design of metaheuristics using reinforcement learning within a novel general search framework. *IEEE Transactions on Evolutionary Computation*, 27(4):1072–1084, 2022.

[138] Babatunji Omoniwa, Boris Galkin, and Ivana Dusparic. Communication-enabled deep reinforcement learning to optimise energy-efficiency in UAV-assisted networks. *Vehicular Communications*, 43:100640, 2023.

[139] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

[140] Akshat Kumar, Adrian Petcu, and Boi Faltings. H-DPOP: Using hard constraints for search space pruning in DCOP. In *AAAI*, pages 325–330, 2008.

[141] Qi Cai, Zhuoran Yang, and Zhaoran Wang. Reinforcement learning from partial observation: Linear function approximation with provable sample efficiency. In *International Conference on Machine Learning*, pages 2485–2522. PMLR, 2022.

[142] Omkar Tilak and Snehasis Mukhopadhyay. Decentralized and partially decentralized reinforcement learning for distributed combinatorial optimization problems. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 389–394. IEEE, 2010.

[143] Marlos C Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80):1–69, 2023.

[144] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning. In *2020 IEEE conference on games (CoG)*, pages 479–486. IEEE, 2020.

[145] Sultan J Majeed and Marcus Hutter. Exact reduction of huge action spaces in general reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8874–8883, 2021.

[146] Zhangjie Fu, Yuanhang Mao, Daojing He, Jingnan Yu, and Guowu Xie. Secure multi-UAV collaborative task allocation. *IEEE Access*, 7:35579–35587, 2019.

[147] Evşen Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks*, 68:1–15, 2018.

[148] Thi Thoa Mac, Cosmin Copot, Robin De Keyser, and Clara M Ionescu. The development of an autonomous navigation system with optimal control of an UAV in partly unknown indoor environment. *Mechatronics*, 49:187–196, 2018.

[149] David Castells-Graells, Christopher Salahub, and Evangelos Pournaras. On cycling risk and discomfort: urban safety mapping and bike route recommendations. *Computing*, 102(5):1259–1274, 2020.

[150] Jun Tang, Songyang Lao, and Yu Wan. Systematic review of collision-avoidance approaches for unmanned aerial vehicles. *IEEE Systems Journal*, 16(3):4356–4367, 2021.

[151] Chuhao Qin, Alexander Robins, Callum Lillywhite-Roake, Adam Pearce, Hritik Mehta, Scott James, Tsz Ho Wong, and Evangelos Pournaras. M-SET: Multi-drone swarm intelligence experimentation with collision avoidance realism. In *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pages 1–7, 2024. doi: 10.1109/LCN60385.2024.10639825.

[152] Ana Batinovic, Jurica Goricanec, Lovro Markovic, and Stjepan Bogdan. Path planning with potential field-based obstacle avoidance in a 3D environment by an unmanned aerial vehicle. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 394–401. IEEE, 2022.

[153] Zeinab Nezami, Emmanouil Chaniotakis, and Evangelos Pournaras. When computing follows vehicles: Decentralized mobility-aware resource allocation for edge-to-cloud continuum. *IEEE Internet of Things Journal*, 2025.

[154] Zeinab Nezami, Evangelos Pournaras, Amir Borzouie, and Jie Xu. Smotec: An edge computing testbed for adaptive smart mobility experimentation. In *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 1–7. IEEE, 2023.

[155] Gözde Kizilateş and Fidan Nuriyeva. On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in Computational Science, Engineering and Information Technology*, pages 111–118. Springer, 2013.

[156] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. Multi-agent reinforcement learning-based resource allocation for UAV networks. *IEEE Transactions on Wireless Communications*, 19(2):729–743, 2019.

[157] Damian Wierzbicki. Multi-camera imaging system for UAV photogrammetry. *Sensors*, 18(8):2433, 2018.

[158] Jovan Nikolic and Evangelos Pournaras. Structural self-adaptation for decentralized pervasive intelligence. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 562–571. IEEE, 2019.

[159] Novella Bartolini, Andrea Coletta, and Gaia Maselli. On task assignment for early target inspection in squads of aerial drones. In *2019 IEEE 39th International*

Conference on Distributed Computing Systems (ICDCS), pages 2123–2133. IEEE, 2019.

[160] Majed Alwateer and Seng W Loke. A two-layered task servicing model for drone services: Overview and preliminary results. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 387–390. IEEE, 2019.

[161] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. Energy and delay aware task assignment mechanism for UAV-based IoT platform. *IEEE Internet of Things Journal*, 6(4):6523–6536, 2019.

[162] Novella Bartolini, Andrea Coletta, Gaia Maselli, et al. A multi-trip task assignment for early target inspection in squads of aerial drones. *IEEE Transactions on Mobile Computing*, 20(11):3099–3116, 2020.

[163] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research*, 36(2):231–258, 2017.

[164] Ilias Gerostathopoulos and Evangelos Pournaras. Trapped in traffic? A self-adaptive framework for decentralized traffic optimization. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 32–38. IEEE, 2019.

[165] Yong Sik Chang and Hyun Jung Lee. Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104: 307–317, 2018.

[166] Meituan. Meituan drone launches the fourth generation of new models and exhibits new urban low-altitude logistics solutions at 2023 waic, 2023. Available: https://www.meituan.com/news/NN230706019014042.

[167] EdeSeven. Britain's electricity generation - december 2024, 2025. Available: https://www.edenseven.co.uk/.

[168] Chase C Murray and Amanda G Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.

[169] Petr Stodola and Libor Kutěj. Multi-depot vehicle routing problem with drones: Mathematical formulation, solution algorithm and experiments. *Expert Systems with Applications*, 241:122483, 2024.

[170] Andy M Ham. Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91:1–14, 2018.

[171] Stefan Poikonen, Xingyin Wang, and Bruce Golden. The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43, 2017. ISSN 1097-0037. doi: 10.1002/net.21746.

[172] Byung Duk Song, Kyungsu Park, and Jonghoe Kim. Persistent uav delivery logistics: MILP formulation and efficient heuristic. *Computers & Industrial Engineering*, 120:418–428, 2018.

[173] Bin Liu, Wei Ni, and Hongbo Zhu. Optimal charging scheduling and speed control for delay-bounded drone delivery. *IEEE Transactions on Vehicular Technology*, 2024.

[174] Wen-Chyuan Chiang, Yuyu Li, Jennifer Shang, and Timothy L Urban. Impact of drone delivery on sustainability and cost: Realizing the uav potential through vehicle routing optimization. *Applied energy*, 242:1164–1175, 2019.

[175] Mohammad Sadra Rajabi, Pedram Beigi, and Sina Aghakhani. Drone delivery systems and energy management: A review and future trends. *Handbook of smart energy systems*, pages 1–19, 2023.

[176] Youngmin Choi and Paul M Schonfeld. A comparison of optimized deliveries by drone and truck. *Transportation Planning and Technology*, 44(3):319–336, 2021.

[177] Batool Madani and Malick Ndiaye. Hybrid truck-drone delivery systems: A systematic literature review. *IEEE Access*, 2022.

[178] Arun Narayanan, Evangelos Pournaras, and Pedro HJ Nardelli. Large-scale package deliveries with unmanned aerial vehicles using collective learning. *IEEE Intelligent Systems*, 2024.

[179] Stefan Poikonen and Bruce Golden. Multi-visit drone routing problem. *Computers & Operations Research*, 113:104802, 2020.

[180] Zhihao Luo, Mark Poon, Zhenzhen Zhang, Zhong Liu, and Andrew Lim. The multi-visit traveling salesman problem with multi-drones. *Transportation Research Part C: Emerging Technologies*, 128:103172, 2021.

[181] Xin Wang, Jiemin Zhao, Chun Cheng, and Mingyao Qi. A multi-objective fuzzy facility location problem with congestion and priority for drone-based emergency deliveries. *Computers & Industrial Engineering*, 179:109167, 2023.

[182] Juan Zhang, James F. Campbell, Donald C. Sweeney II, and Andrea C. Hupman. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment*, 90:102668, January 2021. ISSN 13619209. doi: 10.1016/j.trd.2020.102668.

[183] Luigi Di Puglia Pugliese, Francesca Guerriero, and Maria Grazia Scutellá. The last-mile delivery process with trucks and drones under uncertain energy consumption. *Journal of Optimization Theory and Applications*, 191(1):31–67, 2021.

[184] Xia Zhang and Shuang Zeng. The drone-assisted simultaneous pickup and delivery problem with time windows. *Computers & Operations Research*, page 106996, 2025.

[185] Yao Liu, Jianmai Shi, Zhihao Luo, Xingchen Hu, Witold Pedrycz, and Zhong Liu. Cooperated truck-drone routing with drone energy consumption and time windows. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[186] Zhiliang Bi, Xiwang Guo, Jiacun Wang, Shujin Qin, and Guanjun Liu. Truck-drone delivery optimization based on multi-agent reinforcement learning. *Drones*, 8(1):27, 2024.

[187] Fan Wu and Lixia Wu. Deepeta: A spatial-temporal sequential neural network model for estimating time of arrival in package delivery system. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 774–781, 2019.

[188] Menghua Deng, Yuanbo Li, Jianpeng Ding, Yanlin Zhou, and Lianming Zhang. Stochastic and robust truck-and-drone routing problems with deadlines: A benders decomposition approach. *Transportation Research Part E: Logistics and Transportation Review*, 190:103709, 2024.

[189] Xiulan Shu, Anping Lin, and Xupeng Wen. Energy-saving multi-agent deep reinforcement learning algorithm for drone routing problem. *Sensors*, 24(20): 6698, 2024.

[190] Pannee Suanpang and Pitchaya Jamjuntr. Optimizing last-mile delivery by deep q-learning approach for autonomous drone routing in smart logistics. *Operational Research in Engineering Sciences: Theory and Applications*, 7(2), 2024.

[191] Evangelos Pournaras, Atif Nabi Ghulam, Renato Kunz, and Regula Hänggli. Crowd sensing and living lab outdoor experimentation made easy. *IEEE Pervasive Computing*, 21(1):18–27, 2021.

[192] Prithvi Krishna Chittoor, Bharatiraja Chokkalingam, and Lucian Mihet-Popa. A review on UAV wireless charging: Fundamentals, applications, charging techniques and standards. *IEEE Access*, 9:69235–69266, 2021.

[193] Mark V Mamchenko. Analysis of control channel cybersecurity of the consumer-grade UAV by the example of DJI Tello. In *Journal of Physics: Conference Series*, volume 1864, page 012127. IOP Publishing, 2021.

[194] Huai Chuangfeng, Liu Pingan, and Jia Xueyan. Measurement and analysis for lithium battery of high-rate discharge performance. *Procedia Engineering*, 15: 2619–2623, 2011.

[195] Mouhyemen Khan, Karel Heurtefeux, Amr Mohamed, Khaled A Harras, and Mohammad Mehedi Hassan. Mobile target coverage and tracking on drone-be-gone UAV cyber-physical testbed. *IEEE Systems Journal*, 12(4):3485–3496, 2017.

[196] Ouns Bouachir, Moayad Aloqaily, Fabien Garcia, Nicolas Larrieu, and Thierry Gayraud. Testbed of qos ad-hoc network designed for cooperative multi-drone tasks. In *Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access*, pages 89–95, 2019.

[197] Mohammad Reza Rezaee, Nor Asilah Wati Abdul Hamid, Masnida Hussin, and Zuriati Ahmad Zukarnain. Comprehensive review of drones collision avoidance schemes: Challenges and open issues. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[198] Alexandros Kouris and Christos-Savvas Bouganis. Learning to fly by myself: A self-supervised CNN-based approach for autonomous navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[199] Ning Wang, Markus Christen, and Matthew Hunt. Ethical considerations associated with “humanitarian drones”: A scoping literature review. *Science and engineering ethics*, 27(4):51, 2021.

[200] Civil Aviation Authority. Unmanned aircraft and aircraft systems, 2025. Available: https://www.caa.co.uk/.

[201] Graeme Horsman. Unmanned aerial vehicles: A preliminary analysis of forensic challenges. *Digital Investigation*, 16:1–11, 2016.

[202] Kay Wackwitz and Hendrick Boedecker. Safety risk assessment for uav operation. *Drone Industry Insights, Safe Airspace Integration Project, Part One, Hamburg, Germany*, pages 31–53, 2015.

[203] Wardatul Hayat Adnan and Mohd Fadly Khamis. Drone use in military and civilian application: Risk to national security. *Journal of Media and Information Warfare (JMIW)*, 15(1):60–70, 2022.

[204] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*, pages 2681–2690. PMLR, 2017.

[205] Maninderpal Singh, Gagangeet Singh Aujla, and Rasmeet Singh Bali. A deep learning-based blockchain mechanism for secure internet of drones environment. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4404–4413, 2020.

[206] Chuhao Qin, Pan Li, Jun Liu, and Jianqi Liu. Blockchain-enabled charging scheduling for unmanned vehicles in smart cities. *Journal of Internet Technology*, 22(2):327–337, 2021.

[207] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on computer communications*, pages 1698–1707. IEEE, 2020.

[208] Min Hao, Chen Shang, Siming Wang, Wenchao Jiang, and Jiangtian Nie. UAV-assisted zero knowledge model proof for generative AI: A multi-agent deep reinforcement learning approach. *IEEE Internet of Things Journal*, 2025.

[209] Evangelos Pournaras and Jose Espejo-Uribe. Self-repairable smart grids via online coordination of smart transformers. *IEEE Transactions on Industrial Informatics*, 13(4):1783–1793, 2016.

List of Figures

List of Tables

List of Code/Data/Videos

Hierarchical reinforcement and collective learning, which combines multi-agent reinforcement learning (MAPPO) and multi-agent collective learning (I-EPOS): https://github.com/TDI-Lab/HRCL

Constraints modelling in decentralized software, enforcing individual and aggregate level constraints for EPOS project for Collective Learning which is voted in the top 100 project by Unesco IRCAI:
https://github.com/epournaras/EPOS

Multi-Agent Reinforcement learning-based Optimized Plan Selection, a powerful approach to solve vehicle routing problem for multi-drone last-mile delivery: https://github.com/TDI-Lab/MAR-OPS

M-SET: Multi-drone swarm intelligence experimentation with collision avoidance realism and a documentation for guidance:
https://github.com/TDI-Lab/M-SET
https://github.com/TDI-Lab/M-SET-Documentation

DJI Tello Edu UAVs output datasets, recorded by the testbed prototyping: https://doi.org/10.6084/m9.figshare.20069366.v5

EPOS-based plans for multi-drones sensing, generated by the coordination model for spatio-temporal sensing:
https://doi.org/10.6084/m9.figshare.7806548.v6

EPOS with Drones & CDCA, the videos to record the navigation, sensing, recharging and collision avoidance of Crazyflies:
https://m.youtube.com/channel/UCbzdkFvuMy4sJmsXAmfNr0w

Visualization of multi-drone route planning, the videos of multi-drone navigation, sensing and recharging using different approaches:
https://www.youtube.com/channel/UCKkWPVTy36rcN2lUa0YwRPA

Abbreviations

UAV	Unmanned Aerial Vehicle
PMAC	Planning-based Multi-Agent Coordination
OPS	Optimized Plan Selection
HALOP	Hierarchical multi-Agent Learning-based Optimized Planning
M-TET	Multi-drone Tasking Experimentation Testbed
EPOS	Economic Planning and Optimized Selection
I-EPOS	Iterative Economic Planning and Optimized Selection
MARL	Multi-Agent Reinforcement Learning
MAPPO	Multi-Agent Proximal Policy Optimization
HRL	Hierarchical Reinforcement Learning
CBBA	Concensus-Based Bundle Algorithm
RMSE	Root Mean Square Error
PFG	Potential Fields Grid
SUMO	Simulations of Urban MObility

Publications

Chuhao Qin, Fethi Candan, Lyudmila Mihaylova, and Evangelos Pournaras. 3, 2, 1, drones go! a testbed to take off UAV swarm intelligence for distributed sensing. In *UK Workshop on Computational Intelligence*, pages 576-587. Springer, 2022.

Chuhao Qin and Evangelos Pournaras. Coordination of drones at scale: Decentralized energy-aware swarm intelligence for spatio-temporal sensing. *Transportation Research Part C: Emerging Technologies*, 157:104387, 2023.

Srijoni Majumdar, Chuhao Qin, and Evangelos Pournaras. Discrete-choice multi- agent optimization: Decentralized hard constraint satisfaction for smart cities. In International Conference on Autonomous Agents and Multiagent Systems, pages 60-76. Springer, 2023.

Chuhao Qin, Alexander Robins, Callum Lillywhite-Roake, Adam Pearce, Hritik Mehta, Scott James, Tsz Ho Wong, and Evangelos Pournaras. M-SET: Multi-drone swarm intelligence experimentation with collision avoidance realism. In *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pages 1-7. IEEE, 2024.

Chuhao Qin and Evangelos Pournaras. Short vs. long-term coordination of drones: When distributed optimization meets deep reinforcement learning. *arXiv preprint arXiv:2311.09852*, 2023. (**Submitted, Under Review**)

Zeinab Nezami, Zhuolun Li, Chuhao Qin, Fatemeh Banaie, Rabiya Khalid and Evangelos Pournaras. Blockchain and Edge Computing Nexus: A Large-scale Systematic Literature Review. *arXiv preprint arXiv:2506.08636*, 2025. (**Submitted, Under Review**)

Chuhao Qin, Arun Narayanan, and Evangelos Pournaras. Coordinated Multi-Drone Last-mile Delivery: Learning Strategies for Energy-aware and Timely Operations. *arXiv preprint arXiv:2509.15830*, 2025. (**Submitted, Under Review**)

Chuhao Qin and Evangelos Pournaras. Strategic Coordination for Evolving Multi-agent Systems: A Hierarchical Reinforcement and Collective Learning Approach. *arXiv preprint arXiv:2509.18088*, 2025. (**Submitted, Under Review**)