5-31-2021

# Intelligent and secure fog-aided internet of drones

Jingjing Yao
*New Jersey Institute of Technology*

# Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

## ABSTRACT

## INTELLIGENT AND SECURE FOG-AIDED INTERNET OF DRONES

**by**
**Jingjing Yao**

Internet of drones (IoD), which utilize drones as Internet of Things (IoT) devices, deploys several drones in the air to collect ground information and send them to the IoD gateway for further processing. Computing tasks are usually offloaded to the cloud data center for intensive processing. Many IoD applications require real-time processing and event response (e.g., disaster response and virtual reality applications). Hence, data processing by the remote cloud may not satisfy the strict latency requirement. Fog computing attaches fog nodes, which are equipped with computing, storage and networking resources, to IoD gateways to assume a substantial amount of computing tasks instead of performing all tasks in the remote cloud, thus enabling immediate service response. Fog-aided IoD provisions future events prediction and image classification by machine learning technologies, where massive training data are collected by drones and analyzed in the fog node. However, the performance of IoD is greatly affected by drones' battery capacities. Also, aggregating all data in the fog node may incur huge network traffic and drone data privacy leakage.

To address the challenge of limited drone battery, the power control problem is first investigated in IoD for the data collection service to minimize the energy consumption of a drone while meeting the quality of service (QoS) requirements. A PowEr conTROL (PETROL) algorithm is then proposed to solve this problem and its convergence rate is derived.

The task allocation (which distributes tasks to different fog nodes) and the flying control (which adjusts the drone's flying speed) are then jointly optimized to minimize the drone's journey completion time constrained by the drone's battery capacity and task completion deadlines. In consideration of the practical scenario

that the future task information is difficult to obtain, an online algorithm is designed to provide strategies for task allocation and flying control when the drone visits each location without knowing the future.

The joint optimization of power control and energy harvesting control is also studied to determine each drone's transmission power and the transmitted energy from the charging station in the time-varying IoD network. The objective is to minimize the long-term average system energy cost constrained by the drones' battery capacities and QoS requirements. A Markov Decision Process (MDP) is formulated to characterize the power and energy harvesting control process in time-varying IoD networks. A modified actor-critic reinforcement learning algorithm is then proposed to tackle the problem.

To address the challenge of drone data privacy leakage, federated learning (FL) is proposed to preserve drone data privacy by performing local training in drones and sharing training model parameters with a fog node without uploading drone raw data. However, drone privacy can still be divulged to ground eavesdroppers by wiretapping and analyzing uploaded parameters during the FL training process. The power control problem of all drones is hence investigated to maximize the FL system security rate constrained by drone battery capacities and the QoS requirements (e.g., FL training time). This problem is formulated as a non-linear programming problem and an algorithm is designed to obtain the optimum solutions with a low computational complexity.

All proposed algorithms are demonstrated to perform better than existing algorithms by extensive simulations and can be implemented in the intelligent and secure fog-aided IoD network to improve system performances on energy efficiency, QoS, and security.

# INTELLIGENT AND SECURE FOG-AIDED INTERNET OF DRONES

by
Jingjing Yao

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

May 2021

**APPROVAL PAGE**

**INTELLIGENT AND SECURE FOG-AIDED INTERNET OF DRONES**

**Jingjing Yao**

---

Dr. Nirwan Ansari, Dissertation Advisor — Date
Distinguished Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Ashutosh Dutta, Committee Member — Date
Senior Research Scientist, Johns Hopkins University Applied Physics Lab

---

Dr. Abdallah Khreishah, Committee Member — Date
Associate Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Qing Liu, Committee Member — Date
Assistant Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Mengchu Zhou, Committee Member — Date
Distinguished Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

## BIOGRAPHICAL SKETCH

**Author:** Jingjing Yao

**Degree:** Doctor of Philosophy

**Date:** May 2021

**Undergraduate and Graduate Education:**

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2021.
- Master of Science in Information and Communication Engineering,
  University of Science and Technology of China, Hefei, China, 2016.
- Bachelor of Science in Information and Communication Engineering,
  Dalian University of Technology, Dalian, China, 2013.

**Major:** Computer Engineering

**Presentations and Publications:**

**J. Yao** and N. Ansari, "Wireless Power and Energy Harvesting Control in IoD by Deep Reinforcement Learning," *IEEE Transactions on Green Communications and Networking*, doi: 10.1109/TGCN.2021.3049500, early access.

**J. Yao** and N. Ansari, "Enhancing Federated Learning in Fog-Aided IoT by CPU Frequency and Wireless Power Control," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3438-3445, Mar. 2021.

**J. Yao** and N. Ansari, "Caching in Dynamic IoT Networks by Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3268-3275, Mar. 2021.

**J. Yao** and N. Ansari, "Online Task Allocation and Flying Control in Fog-Aided Internet of Drones," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5562-5569, May 2020.

**J. Yao** and N. Ansari, "Task Allocation in Fog-Aided Mobile IoT by Lyapunov Online Reinforcement Learning," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 556-565, Jun. 2020.

**J. Yao** and N. Ansari, "Power Control in Internet of Drones by Deep Reinforcement Learning," *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020, pp. 1-6.

**J. Yao** and N. Ansari, "Fog Resource Provisioning in Reliability-Aware IoT Networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8262-8269, Oct. 2019.

**J. Yao** and N. Ansari, "QoS-Aware Power Control in Internet of Drones for Data Collection Service," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6649-6656, Jul. 2019.

**J. Yao**, T. Han and N. Ansari, "On Mobile Edge Caching," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2525-2553, third quarter 2019.

**J. Yao** and N. Ansari, "QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 167-175, Mar. 2019.

**J. Yao** and N. Ansari, "Caching in Energy Harvesting Aided Internet of Things: A Game-Theoretic Approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3194-3201, Apr. 2019.

**J. Yao** and N. Ansari, "Joint Content Placement and Storage Allocation in C-RANs for IoT Sensing Service," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1060-1067, Feb. 2019.

**J. Yao** and N. Ansari, "Joint Drone Association and Content Placement in Cache-Enabled Internet of Drones," *IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1-6.

**J. Yao** and N. Ansari, "Energy-Aware Task Allocation for Mobile IoT by Online Reinforcement Learning," *IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-6.

**J. Yao** and N. Ansari, "QoS-Aware Rechargeable UAV Trajectory Optimization for Sensing Service," *IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-6.

**J. Yao** and N. Ansari, "QoS-aware Joint BBU-RRH Mapping and User Association in Cloud-RANs," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 881-889, Dec. 2018.

**J. Yao** and N. Ansari, "Reliability-aware Fog Resource Provisioning for Deadline-driven IoT Services," *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

**J. Yao** and N. Ansari, "Joint Caching in Fronthaul and Backhaul Constrained C-RAN," *IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017, pp. 1-6.

**J. Yao**, P. Lu, L. Gong and Z. Zhu, "On Fast and Coordinated Data Backup in Geo-Distributed Optical Inter-Datacenter Networks," *Journal of Lightwave Technology*, vol. 33, no. 14, pp. 3005-3015, Jul. 2015.

P. Lu, L. Zhang, X. Liu, **J. Yao** and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Network*, vol. 29, no. 5, pp. 36-42, 2015.

**J. Yao**, P. Lu and Z. Zhu, "Minimizing disaster backup window for geo-distributed multi-datacenter cloud systems," *IEEE International Conference on Communications (ICC)*, Sydney, 2014, pp. 3631-3635.

*It has become appallingly obvious that our technology has exceeded our humanity.*

Albert Einstein

## ACKNOWLEDGMENT

Firstly, I would like to express my deepest appreciation to my advisor Dr. Nirwan Ansari for his continuous support of my Ph.D. study. Without his guidance, I would not be able to complete my research and this dissertation. Because of his patience, motivation, and immense knowledge, I could not have imagined having a better advisor.

Second, I would like to thank my dissertation committee members Dr. Ashutosh Dutta, Dr. Abdallah Khreishah, Dr. Qing Liu, and Dr. Mengchu Zhou for their insightful comments and suggestions.

Third, I would like to thank the Electrical and Computer Engineering Department at NJIT for its financial support, and the National Science Foundation Grants No. CNS-1814748 and No. CNS-1647170 for their financial support for my research.

My sincere thanks also go to all members in the Advanced Networking Lab including Xueqing Huang, Xiang Sun, Xilong Liu, Liang Zhang, Qiang Fan, Di Wu, Shuai Zhang, Weiqi Liu, Mohammad Arif Hossain, and Abdullah Hossain. With their help and support, I have experienced a wonderful time in both my study and life at the university.

Last but not the least, I would like to thank my parents Mr. Qingsong Yao and Mrs. Juxiang Hong for supporting me spiritually throughout my life.

## TABLE OF CONTENTS

**TABLE OF CONTENTS**
**(Continued)**

**Chapter** **Page**

**TABLE OF CONTENTS**
**(Continued)**

**Chapter** **Page**

## LIST OF FIGURES

## LIST OF FIGURES
## (Continued)

**Figure** **Page**

# CHAPTER 1

# INTRODUCTION

The Internet of Things (IoT) connects billions of sensors and actuators over a distributed environment to provision various applications such as smart city, home and healthcare [1]. Drones, also known as unmanned aerial vehicles (UAVs), have become an emerging technology for disaster investigation, surveillance and environment monitoring [2]. Integrating drones into IoT networks, where drones act as IoT devices, is referred to as Internet of Drones (IoD) and has been exploited in traffic surveillance, object tracking and disaster rescue [3].

One fundamental application of IoD is the sensing service which collects information of the locations of interest by taking pictures and videos. Several computing tasks are then generated by drones and offloaded to the Internet via the IoD gateways for further processing. The computing results are sent back to drones and then reported to the clients [4]. A drone usually follows a pre-determined transit route to visit all locations of interest [3].

The computing tasks of drones are conventionally offloaded to the remote cloud which provides huge computing resources. However, cloud processing incurs a long network latency and hence degrades the user quality of service (QoS). Fog-aided IoT networks have thus been proposed to improve the IoD service performance for time-sensitive services, where fog nodes are deployed close to IoT devices [5, 6]. In practice, fog nodes are attached to the gateways to provide computing resources and process the deadline-driven computing tasks from drones to achieve fast service response.

The biggest challenge of IoD is the limited drone battery capacity owing to its size and weight limitations. To address this challenge, energy efficient IoD

systems should be designed to reduce the energy consumption of drones. The energy consumption of a drone in IoD networks usually consists of the wireless communication energy consumption for wireless data transmission and the propulsion energy consumption for drone's hovering in the air and transitions among different locations. Both two types of energy consumptions should be considered in designing IoD systems.

In order to reduce the energy consumption for wireless data transmission and drone's hovering in the air, task allocation should be well designed. Task allocation determines which computing tasks should be assigned to which fog nodes [1]. The conventional strategy of task allocation is to assign each task to its nearest fog node. This approach may cause some fog nodes overloaded while others underloaded, thus potentially elongating response latencies and violating QoS requirements. A longer task completion time means more energy is required for drone's data transmission and hovering. Therefore, task allocation affects the drone's energy consumption.

In order to reduce the energy consumption for a drone's transitions, adjusting drone's flying speed should be considered. A high flying speed increases the drone's energy consumption for moving the drone forward in transition. Intuitively, the flying speed should be minimized. However, the low flying speed increases the energy for lifting the drone against the force of gravity when the drone flies [7]. Hence, the flying speed should be balanced and controlled to reduce the energy consumption for the drone's transitions.

Adjusting the drone's wireless transmission power can help reduce the energy consumption of transmitting the collected IoD data [8]. It is thus important to investigate the wireless power control. Energy harvesting can also be a good alternative to prolong the lifetime of drone batteries [9]. Energy harvesting may use the ambient energy sources, e.g., solar and wind, to harvest energy. However, the ambient sources may not guarantee the QoS requirements (e.g., minimum data

transmission time) because they are random and uncertain. Hence, controllable energy sources, e.g., radio frequency (RF) signals from a power station, can be considered to supply energy on demand [10]. We utilize a charging station to charge drone batteries to help maintain drones' operations, where the radio signals are sent from the charging station to carry energy in the form of electromagnetic radiation. Then, the energy harvesting device of each drone converts the radio signals to its battery energy [11]. The harvested energy depends on the transmitted energy of the charging station and the path loss between drones and the charging station [12]. Therefore, energy harvesting control, which determines the transmitted energy from the charging station, is important to be investigated.

The dynamic and time-varying IoD networks also pose a great challenge of wireless power control and energy harvesting control. The varying network states (e.g., collected data, battery level, and QoS requirements) at different time epochs require different power control and energy harvesting control policies to achieve the optimum performance. A Markov decision process (MDP) can be utilized to characterize a time-varying IoD network [13]. It is usually difficult to obtain the complete and accurate information of the MDP model in unknown and dynamic IoD networks. We hence design a deep reinforcement learning algorithm to address the MDP model, i.e., the sequential decision-making problem in time-varying IoD networks [14]. Specifically, the reinforcement learning is a learning process of trials and errors that interacts with the network environment by observing network states (i.e., collected data and battery levels) and then taking actions (i.e., determining the wireless power control and energy harvesting control policies). Deep reinforcement learning uses the deep neural networks (DNNs) to approximate the state-action values in measuring the possible system cost brought by each state-action pair [14].

Another challenge of IoD is the data privacy leakage [15,16]. To enable machine learning services in fog-aided IoD networks, the training data collected by drones are

all sent to the fog node to train the machine learning models (e.g., traffic prediction and object recognition models) [17, 18]. However, massive data transmission injects huge network traffic and degrades the QoS because of wireless bandwidth limitations. On the other hand, the data collected by drones may be sensitive and contain private information (e.g., military areas and human faces). Hence, aggregating all data in the fog node may pose the risk of privacy leakage [19].

Federated learning (FL) [20] is proposed to address the challenges of both the bandwidth limitation and privacy leakage in fog-aided IoD networks. Instead of sending the training data to the fog node in the conventional machine learning services, FL enables local training at each drone on its own data and then shares machine learning model parameters with the fog node. In this way, FL learns a shared global model in the fog node by aggregating the local model parameters derived from distributed drone data in a privacy preserved manner. Much wireless bandwidth can also be saved by avoiding the massive wireless data transmission [21].

The FL performance is usually determined by the FL training time which is composed of the local computation time for model training and the wireless transmission time for transmitting the local model parameters [22]. Hence, the FL performance depends on the drone computing resources and the wireless channels between drones and the fog node. There is a tradeoff between the FL training time and drone energy consumption [23]. To reduce the FL training time, more energy is required to reduce the computation time and wireless data transmission time. Therefore, the FL performance is also limited by the drone battery capacities, which is used for local training computation, wireless data transmission, and drone hovering in the air. Drone wireless transmission power determines the wireless transmission time and the energy consumption for wireless data transmission, and hence is an important factor to be investigated in order to improve the FL performance [21].

As compared with conventional machine learning technologies, FL alleviates the privacy concern by local training. However, security concerns still exist because of data eavesdropping. The ground eavesdroppers may wiretap the local model parameters when drones upload them to the fog node [24]. If the uploaded model parameters are inferred by a malicious eavesdropper, they may leak the private information by model inversion attack [24]. It is hence critical to improve the security of FL communications. Security rate is a key metric to measure the security level of wireless communications, and it is defined as the rate of reliably transmitted information without eavesdropping (i.e., the difference of the data rate between a drone to the fog node and a drone to a eavesdropper) [25]. Wireless power control, which determines the data rates to the fog node and eavesdroppers, is an option to improve the FL system security rate [26].

The rest of this dissertation is organized as follows. In Chapter 2, we investigate the QoS-aware power control problem in IoD for the data collection service. Specifically, we aim to optimize the wireless transmission power at each target location of a data collection task with the objective to minimize the drone's energy consumption constrained by QoS requirements. In Chapter 3, we study the joint optimization of task allocation and flying control problem in IoD. Specifically, we provide insights on task allocation and flying control decisions to minimize the whole journey time (during which all locations of interests are visited and all tasks are processed) constrained by the drone's battery capacity and task completion deadline. In Chapter 4, we investigate the wireless power control and energy harvesting control in time-varying IoD networks for the sensing service. Specifically, we try to optimize each drone's wireless transmission power and the transmitted energy from the charging station to each drone at each time epoch with the objective to minimize the long-term average system energy cost constrained by the drone battery capacities and QoS requirements. In Chapter 5, we investigate the power control

problem for federated learning in fog-aided IoD networks to exploit the tradeoff among FL system security rate, FL training time, and drone energy consumption. Specifically, we optimize each drone's wireless transmission power to maximize the system security rate constrained by the FL training time requirement and each drone's battery capacity. Finally, Chapter 6 summarizes the dissertation.

# CHAPTER 2

# QOS-AWARE POWER CONTROL IN INTERNET OF DRONES FOR DATA COLLECTION SERVICE

Drone-aided networks have been investigated in several works. Yang *et al.* [27] studied the 3-Dimensional (3-D) drone placement problem to relieve overload under heterogeneous traffic. Tang *et al.* [28] utilized multiple UAVs to construct a D2D-enabled network in order to enable content sharing and delivery. Gong *et al.* [29] proposed a localization framework for wireless sensor networks and tailored the framework for a drone to conduct field experiments.

Gharibi *et al.* [30] first proposed the IoD architecture and introduced the five conceptual layers in IoD systems. Koubaa and Qureshi [31] proposed an IoD application to track moving objects. Chen and Wang [32] designed an IoD cloud surveillance system where one or more drones are deployed to collect data of interest. These sensed data are sent to a ground control station and then outsourced to the cloud to be analyzed. Yao and Ansari [8] investigated the UAV trajectory optimization problem for the sensing service to minimize the task completion time constrained by UAV battery capacity.

Power control has been investigated in IoT networks. Li *et al.* [33] proposed a novel ECIoT architecture to achieve mass connections and big data processing. Admission control and power control of IoT devices were jointly optimized by a cross-layer dynamic stochastic strategy. Bader and Alouini [34] proposed a power control mechanism in the context of large-scale IoT. An upper bound was derived on the mean transmit powers for a cluster of IoT devices. Yao and Ansari [35] utilized game theory to optimize the power control policy and caching strategy in the cache-enabled energy harvesting aided IoT. However, the above works assumed that the locations of IoT

devices are fixed. The power control in IoD networks has not been readily addressed yet. In this chapter, we investigate the power control problem for IoD networks for a data collection task to minimize a drone's energy consumption constrained by the QoS requirement.

## 2.1 System Model

In our system model (Figure 2.1), data from $N$ target locations are requested by GW. The collected data are then sent to the service provider for further processing. A drone flies within the flying plane at the height of $H$. We consider a data collection task where a drone flies over different target locations to collect data (e.g., images or videos). When the drone visits each target location $i \in \mathcal{N} = \{1, 2, ..., N\}$, it hovers statically above this location until the collected data are completely transmitted to the GW. Then, the drone transits to the next target location $i + 1$ until the task is completed. We characterize the QoS requirement as the minimum average data transmission rate $R_{th}$ [36]. Note that we only consider a representative drone in our system model. When multiple drones are deployed, their transition routes should be carefully designed to avoid collisions and the bandwidth resource allocation with consideration of interferences should be optimized. This multiple-drone scenario will be left as our future work.

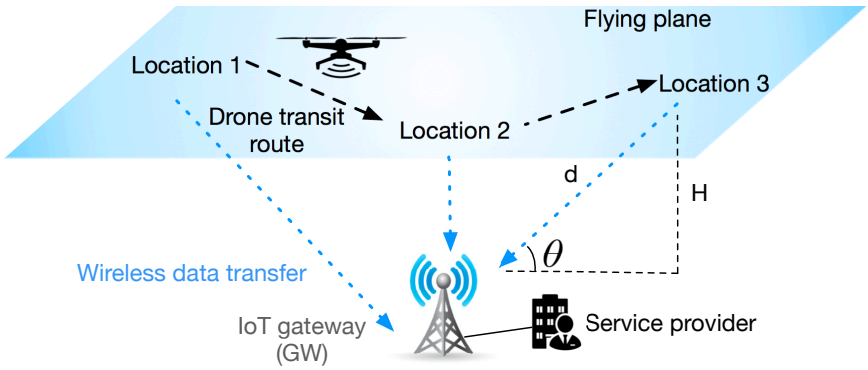



**Figure 2.1** Data collection in IoD.

The air to ground channel model characterizes the channel conditions between the drone and IoT GW. We utilize the widely used probability model [37] which assumes that received signals may be line of sight (LoS) or non-LoS (NLoS) signals with certain probabilities. These probabilities are functions of the environment, density and height of buildings and elevation angle, i.e.,

$$P(LoS) = \frac{1}{1 + \alpha \exp(-\beta[\frac{180}{\pi}\theta - \alpha])}, \tag{2.1}$$

$$P(NLoS) = 1 - P(LoS), \tag{2.2}$$

where $\alpha$ and $\beta$ are environment-related constants (e.g., rural and urban); $\theta$ (as shown in Figure 2.1) is the elevation angle. Note that the probability of receiving the LoS signal increases with a larger elevation angle in Equations (2.1) and (2.2). Then, the average air to ground path loss is

$$\overline{PL} = P(LoS) \times PL_{LoS} + P(NLoS) \times PL_{NLoS}, \tag{2.3}$$

where $PL_{LoS}$ and $PL_{NLoS}$ are the path loss models for LoS and NLoS signals, respectively.

The free space propagation loss and additional excessive path loss values are assigned to their path loss models, which are given by [38]

$$PL_{LoS} = 20 \log_{10}(\frac{4\pi f_c d}{c}) + \xi_{LoS}, \tag{2.4}$$

$$PL_{NLoS} = 20 \log_{10}(\frac{4\pi f_c d}{c}) + \xi_{NLoS}, \tag{2.5}$$

where $f_c$ is the carrier frequency; $d$ is the distance between the IoT GW and drone; $c$ is the speed of light; $\xi_{LoS}$ and $\xi_{NLoS}$ are environment-related constants. Therefore, the wireless transmission rate $R_i$ at location $i$ can be expressed as

$$R_i = W \log_2(1 + \frac{p_i G_i}{N_0 W}), \tag{2.6}$$

where $W$ is the system bandwidth; $p_i$ is the wireless transmission power; $G_i$ denotes the wireless channel gain between the drone at location $i$ and the IoT GW, which is calculated as $G_i = 10^{-\frac{\overline{PL}}{10}}$; $N_0$ denotes the noise power spectrum density.

The energy consumption of a drone usually consists of the propulsion energy to support the drone's transition and hovering in the air, and the communication energy for wireless signal processing and IoT data transmission [7]. The hovering power is generated when the drone remains stationary in the air, which is expressed as [39]

$$P_{hov} = \sqrt{\frac{(mg)^3}{2\pi r_p^2 n_p \rho}}, \tag{2.7}$$

where $m$ is the drone mass; $g$ is the earth gravitational acceleration; $r_p$ is the radius of the drone's propellers; $n_p$ is the number of propellers; $\rho$ is the air density.

The transit power is expressed as [40]

$$P_{trs} = \frac{P_{full}}{v_{full}} v, \tag{2.8}$$

where $P_{full}$ is the hardware power when the drone transits at the full speed of $v_{full}$; $v$ is the drone's current transit speed. Therefore, to complete a data collection task, the propulsion energy consumption of a drone is

$$\begin{aligned} E_{prl} &= \sum_{i=1}^{N-1} P_{trs} T_{i,i+1}^{trs} + \sum_{i=1}^{N} P_{hov} T_i^{com} \\ &= \sum_{i=1}^{N-1} P_{trs} \frac{l_{i,i+1}}{v} + \sum_{i=1}^{N} P_{hov} \frac{D_i}{R_i} \\ &= \sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1} + \sum_{i=1}^{N} P_{hov} \frac{D_i}{R_i}, \end{aligned} \tag{2.9}$$

where $T_{i,i+1}^{trs}$ and $T_i^{com}$ are the transit time between location $i$ and $i+1$, and data transmission time at location $i$, respectively; $l_{i,i+1}$ is the distance between location $i$ and $i+1$; $D_i$ is the data size of the collected data at location $i$.

On the other hand, the drone's communication energy consumption, which is used for transmitting collected IoT data, can be approximated as [41]

$$E_{com} = \sum_{i=1}^{N} (\eta p_i + p_{static}) T_i^{com} = \sum_{i=1}^{N} (\eta p_i + p_{static}) \frac{D_i}{R_i}, \tag{2.10}$$

where $\eta$ is the coefficient of transmission power and $p_{static}$ is the power consumption of drone circuits without data transmission. $R_i$ can be obtained by Equation (5.17). Hence, the total energy consumption of a drone to complete a task is the summation of propulsion energy consumption and communication energy consumption as follows.

$$\begin{aligned} E_{tot} &= E_{prl} + E_{com} \\ &= \sum_{i=1}^{N} (\eta p_i + p_{static} + P_{hov}) \frac{D_i}{R_i} + \sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1} \\ &= \sum_{i=1}^{N} \frac{D_i (\eta p_i + p_{static} + P_{hov})}{W \log_2(1 + \frac{p_i G_i}{N_0 W})} + \sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1}. \end{aligned} \tag{2.11}$$

## 2.2 Problem Formulation

In this section, we formulate the QoS-aware power control problem in IoD networks. Our problem is formulated as

$$\begin{aligned} \textbf{P0:} \min_{p_i} \ & \sum_{i=1}^{N} \frac{D_i (\eta p_i + p_{static} + P_{hov})}{W \log_2(1 + \frac{p_i G_i}{N_0 W})} + \sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1} \\ s.t. \ & \textit{C1: } p_i \le P^m, \ \forall i \in \mathcal{N}, \\ & \textit{C2: } \frac{1}{N} \sum_{i=1}^{N} W \log_2(1 + \frac{p_i G_i}{N_0 W}) \ge R_{th}. \end{aligned}$$

The objective of **P0** is to minimize the total energy consumption $E_{tot}$. Constraint *C1* stipulates that all wireless transmission powers $p_i$ are non-negative and less than the maximum transmission power $P^m$. *C2* implies the QoS requirement which imposes the average transmission rate to be greater than $R_{th}$. Note that **P0** is non-convex and difficult to solve because the objective function is fractional and non-linear.

**Lemma 1.** ***P0*** *is a sum-of-ratios fractional programming problem [42].*

*Proof.* The sum of ratios fractional programming problem is to minimize a sum of fractional functions where the numerator and denominator are convex and concave, respectively. Meanwhile, the constraints are convex sets [42].

The second item of **P0**'s objective function, $\sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1}$ is a constant. The numerator of the first item, $D_i(\eta p_i + p_{static} + P_{hov})$, is linear (i.e., convex) with regard to $p_i$ and the denominator, $W \log_2(1 + \frac{p_i G_i}{N_0 W})$, is concave. Constraint *C1*, $p_i \leq P^m$, is a convex set. In *C2*, $\frac{1}{N} \sum_{i=1}^{N} W \log_2(1 + \frac{p_i G_i}{N_0 W})$ is a concave function with regard to $p_i$ and greater than $R_{th}$. Hence, *C2* is a convex set. Therefore, the lemma is proved. □

The sum-of-ratios problem has been demonstrated to be NP-complete [43] and hence a branch-and-bound algorithm is required to obtain the global optimization solution. However, the branch-and-bound suffers from huge computations and slow convergence. Hence, we propose a better algorithm to solve the problem and demonstrate its convergence rate in next section.

### 2.3 Algorithm Design

In this section, we describe our proposed algorithm (inspired by [44]), PowEr conTROL (PETROL) algorithm, to solve **P0**. Essentially, PETROL is an iteration-based algorithm which, in each iteration, first solves a transformed convex optimization problem by the gradient projection method [45] and then updates the Lagrangian multipliers based on the modified Newton method. The convergence rate of PETROL is also demonstrated.

### 2.3.1 Convex Optimization Transformation

Note that **P0** can be transformed into an equivalent problem **P1** by introducing variables $\omega_i$,

$$\begin{aligned}
\textbf{P1:}\ &\min_{p_i,\omega_i} \sum_{i=1}^{N} \omega_i \\
&s.t.\ C3\text{:}\ \frac{D_i(\eta p_i + p_s)}{W\log_2(1+r_i p_i)} \le \omega_i,\ \forall i \in \mathcal{N}, \\
&\quad C4\text{:}\ p_i \le P^m,\ \forall i \in \mathcal{N}, \\
&\quad C5\text{:}\ \frac{1}{N}\sum_{i=1}^{N} W\log_2(1+r_i p_i) \ge R_{th},
\end{aligned}$$

where the second item of **P0**'s objective function, $\sum_{i=1}^{N-1} \frac{P_{full}}{v_{full}} l_{i,i+1}$, is removed because it is a constant. Note that constraint *C3* can be transformed into $D_i(\eta p_i + p_s) \le w_i W\log_2(1+r_i p_i)$. We also denote that $r_i = \frac{G_i}{N_0 W}$ and $p_s = p_{static} + P_{hov}$ for simplicity. Then, the Lagrangian function of **P1** is

$$\begin{aligned}
\mathcal{L}(\boldsymbol{p}, \boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \vartheta) &= \sum_{i=1}^{N} \omega_i \\
&+ \sum_{i=1}^{N} \lambda_i [D_i(\eta p_i + p_s) - \omega_i W\log_2(1+r_i p_i)] \\
&+ \sum_{i=1}^{N} \mu_i (p_i - P^m) + \vartheta[R_{th} - \frac{1}{N}\sum_{i=1}^{N} W\log_2(1+r_i p_i)],
\end{aligned} \tag{2.12}$$

where $\boldsymbol{\lambda}, \boldsymbol{\mu}$, and $\vartheta$ are the Lagrangian multipliers of constraints *C3*, *C4* and *C5*, respectively.

The Karush-Kuhn-Tucker (KKT) conditions, i.e., Equations (2.13)-(2.20), of **P1** are

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial p_i} =& \lambda_i (D_i \eta - \omega_i W \frac{r_i \log_2 e}{1+r_i p_i}) \\
&+ u_i - \frac{\vartheta W}{N}\frac{r_i \log_2 e}{1+r_i p_i} = 0,\ \forall i \in \mathcal{N},
\end{aligned} \tag{2.13}$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = 1 - \lambda_i W\log_2(1+r_i p_i) = 0,\ \forall i \in \mathcal{N}, \tag{2.14}$$

$$\lambda_i[D_i(\eta p_i + p_s) - \omega_i W \log_2(1 + r_i p_i)] = 0, \ \forall i \in \mathcal{N}, \tag{2.15}$$

$$\mu_i(p_i - P^m) = 0, \ \forall i \in \mathcal{N}, \tag{2.16}$$

$$\vartheta[R_{th} - \frac{1}{N}\sum_{i=1}^{N} W \log_2(1 + r_i p_i)] = 0, \tag{2.17}$$

$$p_i \leq P^m, \ \forall i \in \mathcal{N}, \tag{2.18}$$

$$\frac{1}{N}\sum_{i=1}^{N} W \log_2(1 + r_i p_i) \geq R_{th}, \tag{2.19}$$

$$\lambda_i \geq 0, \mu_i \geq 0, \vartheta \geq 0, \ \forall i \in \mathcal{N}. \tag{2.20}$$

Equation (2.14) is equivalent to

$$\lambda_i = \frac{1}{W \log_2(1 + r_i p_i)} \neq 0, \forall i \in \mathcal{N}, \tag{2.21}$$

which is substituted into Equation (2.15) to yield

$$D_i(\eta p_i + p_s) - \omega_i W \log_2(1 + r_i p_i) = 0, \ \forall i \in \mathcal{N}. \tag{2.22}$$

Note that Equation (2.13) and Equations (2.16)-(2.20) are the KKT conditions of problems **P2** if $\lambda_i$ and $\omega_i$ are fixed.

$$\textbf{P2:} \min_{p_i} \sum_{i=1}^{N} \lambda_i [D_i(\eta p_i + p_s) - \omega_i W \log_2(1 + r_i p_i)]$$

$$s.t.\ C6:\ p_i \leq P^m,\ \forall i \in \mathcal{N},$$

$$C7:\ \frac{1}{N} \sum_{i=1}^{N} W \log_2(1 + r_i p_i) \geq R_{th}.$$

We can observe that problem **P2** is a convex optimization problem. Therefore, the solution of **P0** can be obtained by solving the convex optimization problem **P2** such that Equations (2.21) and (2.22) hold.

Denote $\boldsymbol{x} = [\boldsymbol{\omega}; \boldsymbol{\lambda}] \in \mathbb{R}^{2N}$ and $p_i(\boldsymbol{x})$ as the solution of problem **P2** for a fixed $\boldsymbol{x}$. To satisfy Equations (2.21) and (2.22), we have

$$\begin{cases} -D_i(\eta p_i(\boldsymbol{x}) + p_s) + \omega_i W \log_2(1 + r_i p_i(\boldsymbol{x})) = 0,\ \forall i \in \mathcal{N}, \\ -1 + \lambda_i W \log_2(1 + r_i p_i(\boldsymbol{x})) = 0,\ \forall i \in \mathcal{N}. \end{cases} \tag{2.23}$$

Let $\phi_i(\boldsymbol{x}) = -D_i(\eta p_i(\boldsymbol{x}) + p_s) + \omega_i W \log_2(1 + r_i p_i(\boldsymbol{x})),\ \forall i \in \mathcal{N}$ and $\phi_{N+i}(\boldsymbol{x}) = -1 + \lambda_i W \log_2(1 + r_i p_i(\boldsymbol{x})),\ \forall i \in \mathcal{N}$. Thus, we have

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), ..., \phi_N(\boldsymbol{x}), ..., \phi_{2N}(\boldsymbol{x})]^T = 0. \tag{2.24}$$

Therefore, the solution of **P0** can be considered as the solution of **P2** such that Equation (2.24) holds.

**Lemma 2.** $\boldsymbol{\phi}(\boldsymbol{x})$ *is strongly monotone.*

*Proof.* Let $f_i(\boldsymbol{x}) = W\log_2(1+r_ip_i(\boldsymbol{x})) > 0$ for simplicity. Based on Equations (2.23) and (2.24), the Jacobian matrix of $\boldsymbol{\phi}(\boldsymbol{x})$ can be calculated as

$$\boldsymbol{\phi}'(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & f_N(\boldsymbol{x}) & 0 & \dots & 0 \\ 0 & 0 & 0 & f_1(\boldsymbol{x}) & \dots & \vdots \\ \vdots & \dots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & f_N(\boldsymbol{x}) \end{bmatrix}. \tag{2.25}$$

Since $f_i(\boldsymbol{x}) > 0,\ \forall i \in \mathcal{N}$, we have $\boldsymbol{\phi}'(\boldsymbol{x})$ is a positive definite matrix. Therefore, $\boldsymbol{\phi}(\boldsymbol{x})$ is strongly monotone. □

### 2.3.2 Gradient Projection Method

We first utilize the gradient projection method (GPM) to solve the convex optimization problem **P2**. The Lagrangian function of **P2** is

$$\begin{aligned} \mathcal{L}(\boldsymbol{p},\vartheta) &= \sum_{i=1}^{N}\lambda_i[D_i(\eta p_i + p_s) - \omega_i W\log_2(1+r_ip_i)] \\ &+ \vartheta(R_{th} - \frac{1}{N}\sum_{i=1}^{N} W\log_2(1+r_ip_i)) \\ &= \sum_{i=1}^{N}\lambda_i D_i(\eta p_i + p_s) \\ &- \sum_{i=1}^{N}(\lambda_i w_i + \frac{\vartheta}{N})W\log_2(1+r_ip_i) + \vartheta R_{th}. \end{aligned} \tag{2.26}$$

Hence, the Lagrangian dual function is

$$g(\vartheta) = \inf_{p_i \le P^m} \mathcal{L}(\boldsymbol{p},\vartheta). \tag{2.27}$$

The derivative of $\mathcal{L}(\boldsymbol{p},\vartheta)$ with regard to $\boldsymbol{p}$ is $\frac{\partial\mathcal{L}(\boldsymbol{p},\vartheta)}{\partial p_i} = \lambda_i D_i\eta - (\lambda_i w_i + \frac{\vartheta}{N})\frac{Wr_i\log_2 e}{1+r_ip_i}$. Hence, $\mathcal{L}(\boldsymbol{p},\vartheta)$ is firstly monotonically decreasing and then monotonically increasing.

Meanwhile, $p_i \leq P^m$ should also be satisfied. Therefore, the optimum solution of minimizing $\mathcal{L}(\boldsymbol{p}, \vartheta)$ with regard to $\boldsymbol{p}$ is

$$p_i^* = \min\{P^m, \frac{(\lambda_i w_i + \frac{\vartheta}{N})W \log_2 e}{\lambda_i D_i \eta} - \frac{1}{r_i}\}. \tag{2.28}$$

Substituting Equation (2.28) into Equation (2.27), we have $g(\vartheta) = \mathcal{L}(\boldsymbol{p}^*, \vartheta)$. Hence, the dual problem of **P2** is

$$\begin{aligned} \textbf{P3:} \ & \max_{\vartheta} \ g(\vartheta) = \mathcal{L}(\boldsymbol{p}^*, \vartheta) \\ & s.t. \ C8: \vartheta \geq 0. \end{aligned}$$

The dual problem **P3** can then be solved by the gradient projection method which updates the Lagrangian variable $\vartheta$ based on the gradient search as follows.

$$\vartheta^k = \max\{0, \vartheta^{k-1} + \delta^k(R_{th} - \frac{1}{N}\sum_{i=1}^{N} W \log_2(1 + r_i p_i^*))\}, \tag{2.29}$$

where $k$ is the number of iterations; $\delta^k$ is the step size; $p_i^*$ is defined in Equation (2.28).

### 2.3.3 Modified Newton Method

In order to satisfy Equation (2.24), we utilize the modified Newton method to update $\boldsymbol{x}$. The iterative equation is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta_k \boldsymbol{\tau}_k, \tag{2.30}$$

where $k$ is the number of iterations; $\boldsymbol{\tau}_k$ can be considered as the iterative direction, which is defined as

$$
\begin{aligned}
\boldsymbol{\tau}_k &= -[\boldsymbol{\phi}'(\boldsymbol{x}_k)]^{-1}\boldsymbol{\phi}(\boldsymbol{x}_k) \\
&= -\begin{bmatrix}
\frac{1}{f_1(\boldsymbol{x})} & \dots & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \dots & \vdots \\
0 & \dots & \frac{1}{f_N(\boldsymbol{x})} & 0 & \dots & 0 \\
0 & 0 & 0 & \frac{1}{f_1(\boldsymbol{x})} & \dots & \vdots \\
\vdots & \dots & \vdots & \vdots & \ddots & 0 \\
0 & 0 & 0 & 0 & \dots & \frac{1}{f_N(\boldsymbol{x})}
\end{bmatrix}
\begin{bmatrix}
\phi_1(\boldsymbol{x}) \\ \vdots \\ \phi_N(\boldsymbol{x}) \\ \phi_{N+1}(\boldsymbol{x}) \\ \vdots \\ \phi_{2N}(\boldsymbol{x})
\end{bmatrix} \\
&= -\begin{bmatrix}
\frac{\phi_1(\boldsymbol{x})}{f_1(x)} \\ \vdots \\ \frac{\phi_N(\boldsymbol{x})}{f_N(x)} \\ \frac{\phi_{N+1}(\boldsymbol{x})}{f_1(x)} \\ \vdots \\ \frac{\phi_{2N}(\boldsymbol{x})}{f_N(x)}
\end{bmatrix} = \begin{bmatrix}
\frac{D_1(\eta p_1(\boldsymbol{x})+p_s)}{W\log_2(1+r_1p_1(\boldsymbol{x}))} - \omega_1 \\ \vdots \\ \frac{D_N(\eta p_N(\boldsymbol{x})+p_s)}{W\log_2(1+r_Np_N(\boldsymbol{x}))} - \omega_N \\ \frac{1}{W\log_2(1+r_1p_1(\boldsymbol{x}))} - \lambda_1 \\ \vdots \\ \frac{1}{W\log_2(1+r_Np_N(\boldsymbol{x}))} - \lambda_N
\end{bmatrix};
\end{aligned}
\tag{2.31}
$$

$\delta_k \in (0,1)$ is the iterative step size, which satisfies

$$
\|\boldsymbol{\phi}(\boldsymbol{x}_k + \delta_k\boldsymbol{\tau}_k)\| \leq (1-\epsilon\delta_k)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|, \epsilon \in (0,1). \tag{2.32}
$$

Our proposed algorithm (PETROL) to solve **P0** is then delineated in Algorithm 1. Lines 3-10 iteratively calculate the optimum solution given $\boldsymbol{x}$. In each iteration, Line 4 calculates the solution of the convex problem **P2**. If this solution satisfies Equation (2.24), it is the optimum solution. Otherwise, $\boldsymbol{x}$ is updated according to the modified Newton method, as shown in Lines 8-9.

**Algorithm 1:** PETROL

**Input :** $C, N, W, H_i, N_0, D_i, P^T, P^m, \lambda_i, u_i, v_i$

**Output:** Value $a$ satisfying $\varphi(a) = 0$

Initialize $\epsilon, \boldsymbol{\omega}^1, \boldsymbol{\lambda}^1, \boldsymbol{x}^1 = [\boldsymbol{\omega}^1; \boldsymbol{\lambda}^1]$ ;

Initialize the number of iteration $k = 1$ ;

**while** 1 **do**

  Calculate the solution $\boldsymbol{p}^k(\boldsymbol{x}^k)$ of the convex problem **P2** with fixed $\boldsymbol{\omega}^k, \boldsymbol{\lambda}^k$ by gradient projection method in Equation (2.29) ;

  **if** $\boldsymbol{\phi}(\boldsymbol{x}^k) = 0$ **then**

    **break** ;

  **else**

    Calculate $\delta_k$ satisfying Equation (2.32) ;

    Update $\boldsymbol{x}^{k+1}$ according to Equation (2.30) ;

  **end**

  $k = k + 1$ ;

**end**

**return** $\boldsymbol{p}^k(\boldsymbol{x}^k)$;

**Lemma 3.** *Assume $p_i(\boldsymbol{x})$ $(\forall i \in \mathcal{N})$ satisfies the Lipschitz condition, then PETROL achieves a linear convergence rate to the optimum solution and a quadratic convergence rate in the neighborhood of the optimum solution.*

*Proof.* See Appendix A. □

### 2.4 Performance Evaluation

In this section, we set up simulations to evaluate the performance of our proposed algorithm PETROL. We compare PETROL with the existing work [46] where a data collection task is considered with a fixed wireless transmission power (denoted as NoPowerControl). We also utilize the heuristic algorithm (denoted as Heuristic) for comparison purposes. In Heuristic, the transmission power increases when the wireless channel condition is bad (e.g., the air-to-ground path loss between the drone and IoT gateway is greater than 100 dB) and vice versa. Specifically, at each location, the wireless transmission power is set as the minimal value to satisfy the QoS requirement. We consider a 1000 $m$ $\times$ 1000 $m$ area, where 30 target locations' data are required to be collected and IoT GW is located in the center of this area. These locations are uniformly distributed in this area. The UAV flies at a fixed flying plane at the height of 500 $m$. $\alpha$ and $\beta$ in Equation (2.1) are set to 9.6 and 0.28, respectively. The speed of light $c = 3 \times 10^8$ $m/s$. The carrier frequency $f_c = 2$ $GHz$. $\xi_{LoS}$ and $\xi_{NLoS}$ in Equations (2.4) and (2.5) are 1 $dB$ and 20 $dB$, respectively [37]. The system bandwidth $W = 10$ $MHz$. The drone mass $m = 500$ $gram$, and the earth gravitational acceleration $g = 9.8$ $m/s^2$. $r_p, n_p$ and $\rho$ in Equation (2.7) are 20 $cm$, 4, and 1.225 $kg/m^3$, respectively. The hardware power $P_{full} = 5$ $W$ when the drone transits at the full speed $v_{full} = 15$ $m/s$ [40]. $\eta$ and $p_{static}$ in Equation (2.10) are 4.2 and 8 $W$, respectively. The system bandwidth $W = 10$ $MHz$ and the noise power density $N_0 = -174$ $dBm/Hz$. The amounts of collected data at different locations $D_i$ are uniformly distributed from 1 $MB$ to 10 $MB$. The maximum wireless transmission power $P^m = 5$ $W$ and the QoS

requirement $R_{th} = 135$ *Mbps*. Note that the above values are default values and hence may change as needed. We also plot 90% confidence intervals of the results in our simulations.

Figure 2.2 evaluates the energy consumption of PETROL with different numbers of target locations. Note that since the transit energy consumption (which is a constant) is much larger than the hovering and wireless communication energy consumption, we do not include the transit energy consumption in all figures in order to explicitly show the differences of different algorithms (i.e., the energy consumption is the optimal value of the equivalent problem **P1**). In Figure 2.2, the energy consumptions of PETROL, Heuristic and NoPowerControl all become larger when the number of locations increases. More locations imply that more data should be sent to IoT GW and hence more energy is consumed. PETROL performs the best among the three algorithms and NoPowerControl the worst. NoPowerControl does not adjust the wireless transmission power and hence suffers from longer data transmission time when the wireless channel condition is bad, which results in more energy consumption. Heuristic adjusts the wireless transmission power to satisfy the QoS requirement at each location. However, it neglects the impact of the data size on the energy consumption.

Figure 2.3 compares the performances of PETROL, Heuristic and NoPowerControl with different average data sizes. Note that a larger data size results in more energy consumption for all of PETROL, Heuristic and PETROL because a larger data size requires more data transmission time and hence increases the energy consumption. PETROL requires the least energy consumption and NoPowerControl the most for the similar reason in Figure 2.2. Note that when the data size is less than 30 *Mb*, Heuristic and PETROL achieve approximately the same energy consumptions. However, Heuristic incurs a much lower computational complexity than that of

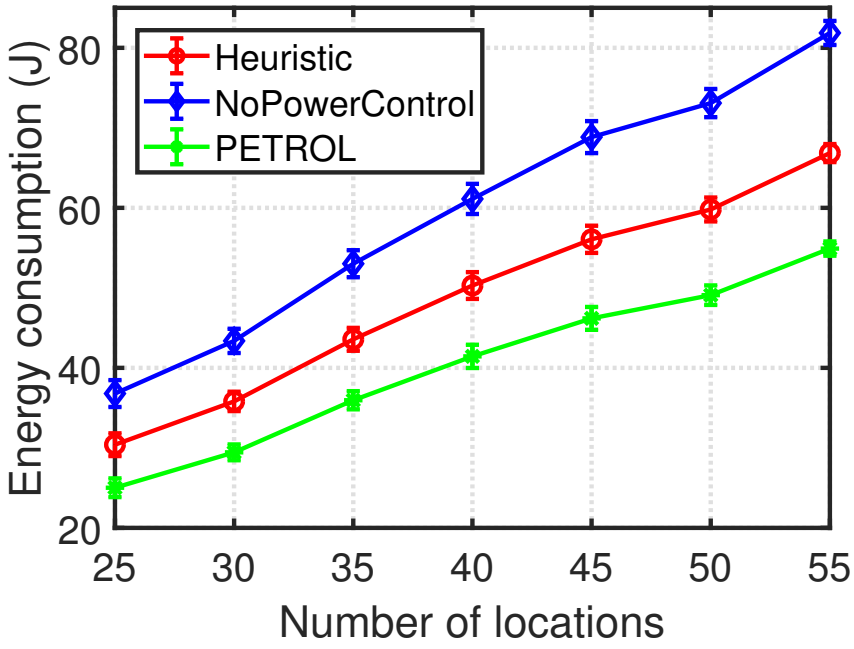



**Figure 2.2** Energy consumption vs number of locations.

PETROL. Hence, adopting Heuristic is beneficial when the data size is less than 30 *Mb*.

Figure 2.4 investigates impacts of different maximum transmission power $P^m$ on the performances of all algorithms. The energy consumptions of PETROL, Heuristic and NoPowerControl all go up because a larger $P^m$ allows the drone to transmit data with larger transmission power and hence increases the energy consumption. PETROL always performs better than Heuristic and NoPowerControl. Note that the differences between PETROL and its comparison algorithms (i.e., Heuristic and NoPowerControl) become larger when $P^m$ increases. This is because when $P^m$ is small, all algorithms adopt the maximum transmission power in order to satisfy the QoS requirement and hence their energy consumptions are similar.

Figure 2.5 illustrates the energy consumptions with different QoS requirements $R_{th}$. A larger $R_{th}$ implies that more transmission power is required and hence increases the energy consumption. Therefore, in Figure 2.5, the energy consumptions

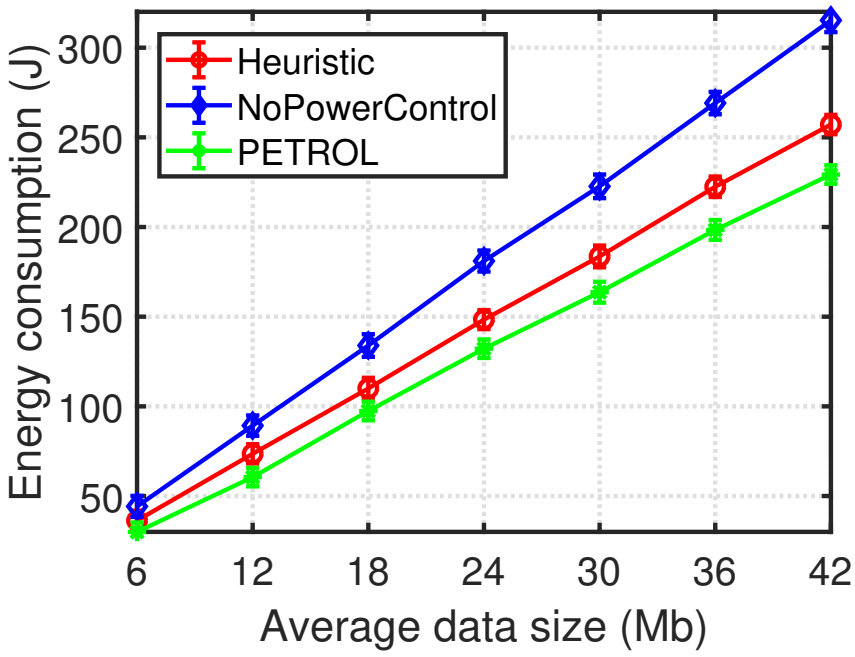



**Figure 2.3** Energy consumption vs average data size.

of PETROL, Heuristic and NoPowerControl all increase as $R_{th}$ increases. PETROL performs the best among the three algorithms. Note that the energy consumptions of all three algorithms remain the same when $R_{th}$ arises from 150 to 160 *Mbps* because of the limitation of $P^m$. On that occasion, increasing $R_{th}$ does not affect the energy consumption because the transmission power has reached its upper limit.

The convergence rates of GPM and PETROL are depicted in Figure 2.6. GPM, denoting the gradient projection method, solves the transformed convex optimization problem in PETROL. Figure 2.6(a) illustrates how $\vartheta$ converges as the number of iterations increases. $\vartheta$ is the optimal solution of dual problem **P3** which is calculated by GPM in order to solve the convex optimization problem **P2**. We can observe that $\vartheta$ converges to a fixed value after more than 300 iterations. PETROL utilizes the modified Newton method to update the parameters in order to satisfy $\boldsymbol{\phi(x)}$. In Figure 2.6(b), function $\sum_{i=1}^{N} \boldsymbol{\phi_i(x_i)}$ converges to 0 after 250 iterations.

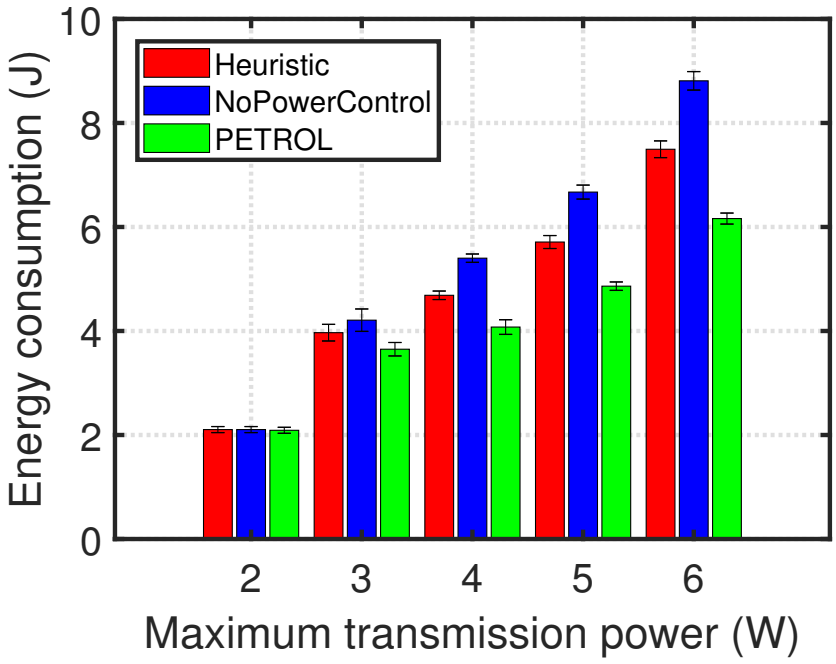



**Figure 2.4** Energy consumption vs maximum transmission power.

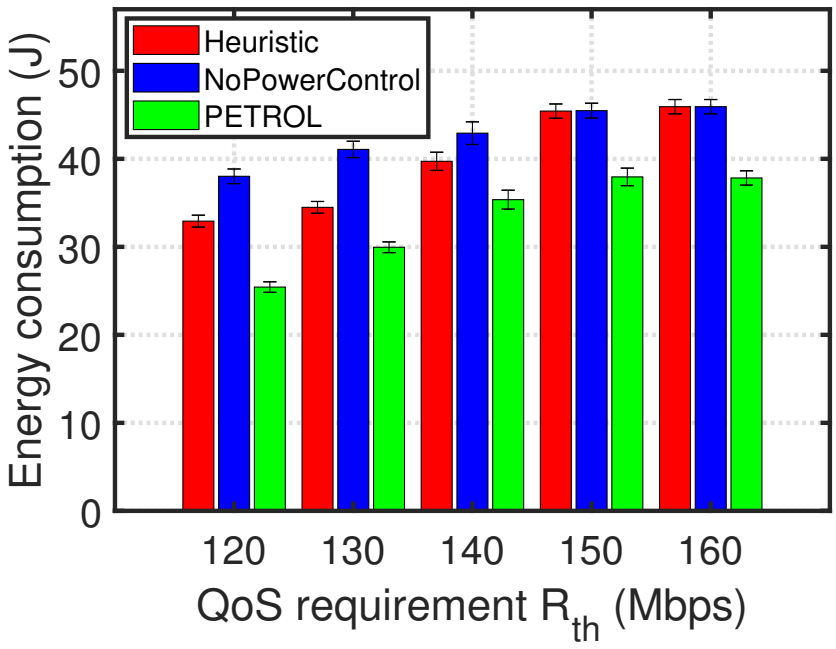
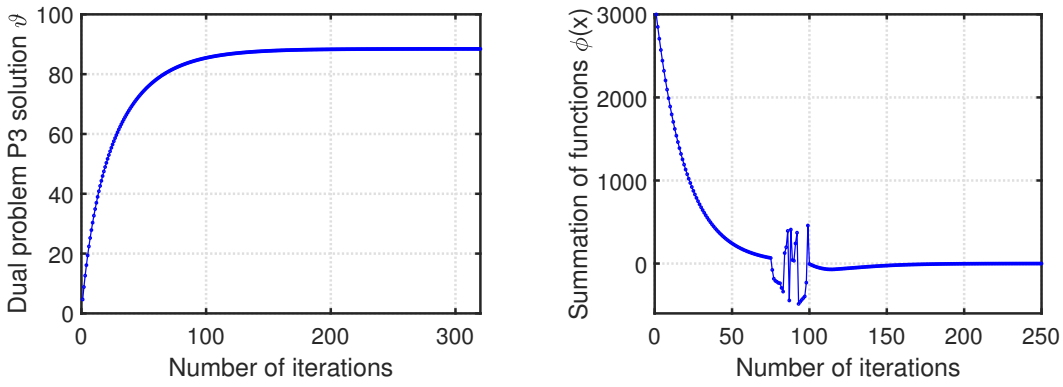



**Figure 2.5** Energy consumption vs QoS requirement.




(a) GPM

(b) PETROL

**Figure 2.6** Convergence rate of GPM and PETROL.

## 2.5 Summary

In this chapter, the power control problem in IoD has been investigated. A sum-of-ratios fractional programming problem has been formulated to minimize the drone's energy consumption constrained by the maximum wireless transmission power and QoS requirements. In order to solve this NP-complete problem, an iteration-based algorithm (PETROL) has been designed; it first obtains the optimal solution of a transformed convex optimization problem by a gradient projection method and then updates the Lagrangian parameters by a modified Newton method. The convergence rate of PETROL has also been proved to achieve a linear rate to the optimum solution and a quadratic rate in the neighborhood of the optimum solution. Simulation results have demonstrated PETROL performs better than the existing algorithms.

# CHAPTER 3

# ONLINE TASK ALLOCATION AND FLYING CONTROL IN FOG-AIDED INTERNET OF DRONES

Drones have been deployed for various applications. Yang *et al.* [27] investigated the 3-D drone-cell placement problem to alleviate the traffic congestion in cellular networks. Tang *et al.* [28] adopted drones to promptly construct the D2D-enabled wireless network. They utilized game theory to solve the channel assignment problem. Yao and Ansari [8] optimized the drone's trajectory to minimize the network delay for the sensing service constrained by the drone's battery capacity. Dorling *et al.* [47] optimized the battery and payload weight for package delivery to minimize the delivery time.

Gharibi *et al.* [30] first proposed the IoD architecture and the conceptual layers in IoD systems. Koubaa and Qureshi [31] proposed a real-time object tracking system in IoD networks. In order to reduce the energy consumption of drones, Yao and Ansari [3] investigated the power control problem in IoD networks for the data collection service in order to minimize the drone's energy consumption constrained by the QoS requirement. Zeng and Zhang [7] optimized the drone's trajectory by jointly considering the network communication throughput and drone's energy consumption in order to maximize the network energy efficiency.

Task allocation has been studied in IoT networks. Zeng *et al.* [48] investigated the task scheduling and resource management strategy in fog-aided networks to minimize the task completion time. Deng *et al.* [49] formulated a workload allocation problem to balance the tradeoff between power consumption and transmission delay in a fog-cloud computing system, where tasks are allocated to both fog nodes and cloud in order to minimize the power consumption.

None of the above works consider the joint optimization of task allocation and flying control (which adjusts the flying speed) in IoD networks. Hence, we investigate the task allocation and flying control to minimize the whole journey time of visiting all locations of interest and processing all tasks by considering the drone's battery capacity and task completion deadline in fog-aided IoD networks.

## 3.1 System Model

In our system model, as shown in Figure 3.1, a drone is launched in the flying plane at a certain height of $H$. In practice, $H$ is usually chosen as the minimum altitude to avoid all obstructions [8]. The drone aims to complete a journey over $M$ locations of interest. We denote the set of all indexes of locations of interest as $\mathcal{M} = \{1, ..., M\}$. The drone starts from location 1, then visits location 2 to location $M$ sequentially, and finally ends at location 0. We assume that the drone flies in a straight line in each segment between any two locations, and may change its heading direction and adjust its flying speed between different segments [7]. We denote the flying speed of segment $i$ (starting from location $i$ to location $i + 1$) as $v_i$. When the drone arrives at each location, it hovers above this location, collects information (e.g., images and videos) over the area, generates several tasks, and then offloads all tasks to the fog nodes for processing. After the drone receives the computing results from the fog nodes, it traverses to the next location and repeats the process until it completes the whole journey to location 0.

At location $i$, the drone generates $K_i$ computing tasks. We characterize each task $k$ as a three-parameter model $< l_{ik}, \vartheta_{ik}, D_{ik} >$ [50], where $l_{ik}$ is the data length in bits, $\vartheta_{ik}$ is the computing intensity (which converts data length into CPU cycles) in CPU cycles per bit, and $D_{ik}$ is the task completion deadline. The task information (i.e., task length, computing intensity, and minimum task processing deadline) is first sent to the IoT service manager which has the knowledge of all

network status information (e.g., computing capacity of each fog node, and wireless channel conditions between all fog nodes and the drone). The service manager then makes the task allocation decisions on which task should be processed at which fog node based on the task information and network status information. After the task allocation decisions are made, the service manager notifies the drone to offload its tasks to the corresponding serving fog nodes which then process the tasks and send the computing results back to the drone. We assume there are $N$ fog nodes in the IoD network and denote the set of indexes of all fog nodes as $\mathcal{N} = \{1, ..., N\}$. Each fog node is attached with an IoT gateway which receives and sends signals to communicate with other network entities (e.g., drone and service manager). Our system model is applicable to several applications. For example, a drone can be deployed to fly over several locations for the disaster recovery application. When the drone arrives at each location, it takes several pictures and videos of this location, and generates several computing tasks. All tasks are then offloaded to the fog nodes to determine whether any people and how many people are there under this location. If the computing results from the fog nodes indicate there are people below, the drone notifies the rescue team to rescue these people.

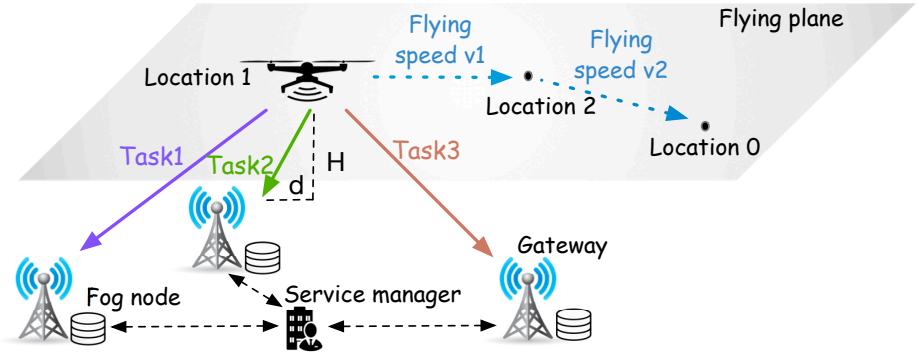



**Figure 3.1** Task allocation and flying speed control in IoD networks.

### 3.1.1 Drone Wireless Transmission Rate

The drone's wireless transmission rate $r_{ij}$ between location $i$ and fog node $j$ is given by [35]

$$r_{ij} = W \log_2(1 + \frac{p_i G_{ij}}{N_0 W}), \tag{3.1}$$

where $W$ is the system bandwidth; $p_i$ is drone $i$'s wireless transmission power; $G_{ij} = 10^{-\frac{\overline{PL}}{10}}$, where $\overline{PL}$ is obtained from Equation (2.3), is the wireless channel gain between location $i$ and fog node $j$; $N_0$ is the noise power spectrum density. Hence, the wireless data transmission delay for offloading task $k$ is

$$d_{ik}^w = \sum_{j=1}^{N} \frac{l_{ik}}{r_{ij}} x_{ijk}, \tag{3.2}$$

where $\frac{l_{ik}}{r_{ij}}$ is the wireless transmission time of task $k$ to fog node $j$, and $x_{ijk} \in \{0, 1\}$ is a binary variable to indicate whether task $k$ at location $i$ is processed at fog node $j$ ($x_{ijk} = 1$ if affirmative).

We denote the computing capacity of fog node $j$ as $f_j$ in CPU cycles per second. Then, the fog processing delay of task $k$ at location $i$ is

$$d_{ik}^c = \sum_{j=1}^{N} \frac{l_{ik} \vartheta_{ik}}{f_j} x_{ijk}. \tag{3.3}$$

Note that the computing result from the fog node is relatively small in size [4], and the downlink transmission rate (from fog nodes to the drone) is usually large. Therefore, we neglect the downlink transmission delay in our work.

The task completion delay $d_{ik}$ of task $k$ at location $i$ can then be expressed as the summation of the uplink wireless transmission delay and the fog processing delay:

$$d_{ik} = d_{ik}^w + d_{ik}^c = \sum_{j=1}^{N} (\frac{l_{ik}}{r_{ij}} + \frac{l_{ik} \vartheta_{ik}}{f_j}) x_{ijk}. \tag{3.4}$$

### 3.1.2 Drone Energy Consumption

The drone's energy consumption usually consists of the propulsion energy (which supports the drone's mobility and hovering in the air) and the energy for wireless data transmissions [7].

We assume the flying speed $v_i$ remains constant within segment $i$ from location $i$ to location $i+1$. The propulsion power from location $i$ to location $i+1$ can then be characterized as a function of $v_i$, and expressed as [7]:

$$P_i^f = c_1 v_i^3 + \frac{c_2}{v_i}, \tag{3.5}$$

where $c_1 v_i^3$ is the required power to move the drone forward, and $\frac{c_2}{v_i}$ is used to lift the drone against the force of gravity; $c_1$ and $c_2$ are constants which are related to the drone's weight and wing area, and the air density. Hence, the propulsion energy consumption for the drone's mobility between location $i$ and $i+1$ is

$$e_i^f = \frac{L_i}{v_i}(c_1 v_i^3 + \frac{c_2}{v_i}) = L_i(c_1 v_i^2 + \frac{c_2}{v_i^2}), \tag{3.6}$$

where $L_i$ and $\frac{L_i}{v_i}$ are the distance and the transition time between location $i$ and $i+1$, respectively.

The drone's propulsion energy consumption for hovering at location $i$ is

$$\begin{aligned} e_i^{hov} &= P_{hov} \sum_{k=1}^{K_i} d_{ik} \\ &= P_{hov} \sum_{k=1}^{K_i} \sum_{j=1}^{N} (\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j}) x_{ijk}, \end{aligned} \tag{3.7}$$

where $P_{hov}$ is a drone's hovering power defined in Equation (2.7).

The drone's energy consumption for transmitting task $k$ to fog node $j$ at location $i$ can be calculated as

$$e_{ijk}^{w} = \frac{p_i l_{ik}}{r_{ij}}, \tag{3.8}$$

where $p_i$ is the drone's wireless transmission power at location $i$ and $\frac{l_{ik}}{r_{ij}}$ is the wireless transmission time of task $k$. Hence, the drone's energy consumption for wireless transmissions at location $i$ is

$$e_{i}^{w} = \sum_{k=1}^{K_i} \sum_{j=1}^{N} \frac{p_i l_{ik}}{r_{ij}} x_{ijk}. \tag{3.9}$$

To sum up all drone's energy consumptions at all locations by incorporating all energy consumptions for wireless data transmission, hovering in the air when the drone offloads tasks to fog nodes, and transitions between different locations, the total energy consumption of drone to complete a whole journey can be expressed as

$$\begin{aligned} e &= \sum_{i=1}^{M} (e_i^w + e_i^{hov} + e_i^f) \\ &= \sum_{i=1}^{M} \sum_{k=1}^{K_i} \sum_{j=1}^{N} [\frac{(p_i + P_{hov}) l_{ik}}{r_{ij}} + \frac{P_{hov} l_{ik} \vartheta_{ik}}{f_j})] x_{ijk} \\ &\quad + \sum_{i=1}^{M} L_i (c_1 v_i^2 + \frac{c_2}{v_i^2}) \end{aligned} \tag{3.10}$$

## 3.2 Problem Formulation

Here, we formulate the problem of jointly optimizing the task allocation and flying speed control in IoD networks. A drone is deployed to collect ground information and generate computing tasks which are offloaded to the fog nodes for processing. To complete a whole journey, the drone needs to travel through all locations of interest and reach the final destination (i.e., location 0). The problem is formulated as

$$\textbf{P0: } \min_{x_{ijk}, v_i} \sum_{i=1}^{M} \sum_{k=1}^{K_i} \sum_{j=1}^{N} \left(\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j}\right) x_{ijk} + \sum_{i=1}^{M} \frac{L_i}{v_i} \tag{3.11}$$

$$\begin{aligned} s.t. \quad & \sum_{j=1}^{N} \left(\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j}\right) x_{ijk} \leq D_{ik}, \\ & \forall i \in \mathcal{M}, k \in \{1, ..., K_i\}, \end{aligned} \tag{3.12}$$

$$\begin{aligned} \sum_{i=1}^{M} \sum_{k=1}^{K_i} \sum_{j=1}^{N} \left[\frac{(p_i + P_{hov}) l_{ik}}{r_{ij}} + \frac{P_{hov} l_{ik} \vartheta_{ik}}{f_j}\right] x_{ijk} \\ + \sum_{i=1}^{M} L_i \left(c_1 v_i^2 + \frac{c_2}{v_i^2}\right) \leq E, \end{aligned} \tag{3.13}$$

$$\sum_{j=1}^{N} x_{ijk} = 1, \ \forall i \in \mathcal{M}, k \in \{1, ..., K_i\}, \tag{3.14}$$

$$\begin{aligned} x_{ijk} \in \{0, 1\}, \\ \forall i \in \mathcal{M}, k \in \{1, ..., K_i\}, j \in \mathcal{N}, \end{aligned} \tag{3.15}$$

$$V_{min} \leq v_i \leq V_{max}, \forall i \in \mathcal{M}. \tag{3.16}$$

The objective in Equation (3.11) is to minimize the whole journey completion time which includes the wireless transmission time, fog processing time at each location, and transition time between different locations. Equation (3.12) imposes the task completion delay of each task $k$ not to exceed the task completion deadline $D_{ik}$. Equation (3.13) implies that the total energy consumption of all tasks is limited by the mobile IoT device battery capacity. Equations (3.14) and (3.15) indicate that each task can only be assigned to one fog node. Equation (3.16) implies that the flying speed in each segment should be within the range of $[V_{min}, V_{max}]$. $V_{min}$ is the

minimum speed to support the drone's mobility while $V_{max}$ is the maximum speed the drone can reach [40].

Note that problem **P0** is a mixed integer non-linear programming (MINLP) problem because the variables $x_{ijk}$ and $v_i$ are binary and continuous, respectively. Obtaining the optimal solution of an MINLP problem is usually difficult because it is not convex [51]. On the other hand, the future information (i.e., future generated tasks at subsequent locations) is difficult to predict and hence unknown to the service manager. Therefore, optimizing problem **P0**, which requires the complete information of the whole journey, might be impracticable in reality. In order to address this problem, we propose an online algorithm to circumvent the requirement of complete information.

## 3.3 Algorithm Design

Owing to the unawareness of future generated task information, we propose an online algorithm to solve the task allocation and flying speed control problem in IoD networks. The difficulty of solving problem **P0** without complete information of tasks at all locations lies in Equation (3.13), where the energy consumptions at different locations are coupled with each other owing to the drone's battery capacity. To be specific, consuming more energy for task allocation at the current location and transition in the current segment will reduce the energy budget for subsequent locations and segments. Therefore, the basic idea of our online algorithm is to break this linkage by introducing the energy deficit $q_i$ of location $i$ and segment $i$ to denote how the energy consumption deviates from the average energy budget.

In our online algorithm, the service manager determines the task allocation strategy for all tasks at location $i$ and the flying speed for segment $i$ without the knowledge of future information at subsequent locations. Since the drone should travel through $M$ locations and segments by consuming less energy than the battery

capacity $E$, the average energy budget for each location and segment is $\frac{E}{M}$. The energy deficit $q_i$ is then defined as

$$q_{i+1} = \max\{q_i + \sum_{k=1}^{K_i}\sum_{j=1}^{N}[\frac{(p_i + P_{hov})l_{ik}}{r_{ij}} + \frac{P_{hov}l_{ik}\vartheta_{ik}}{f_j})]x_{ijk} + L_i(c_1 v_i^2 + \frac{c_2}{v_i^2}) - \frac{E}{M}, 0\}. \tag{3.17}$$

The energy deficit $q_{i+1}$ for location $i+1$ is determined by the energy deficit $q_i$ of location and segment $i$, the energy consumption of location and segment $i$, and the average energy budget. If the energy consumed at location and segment $i$ is larger than the average energy budget (i.e., $\sum_{k=1}^{K_i}\sum_{j=1}^{N}[\frac{(p_i+P_{hov})l_{ik}}{r_{ij}} + \frac{P_{hov}l_{ik}\vartheta_{ik}}{f_j})]x_{ijk} + L_i(c_1 v_i^2 + \frac{c_2}{v_i^2}) > \frac{E}{M}$), an energy deficit is incurred and should be added to the previous deficit $q_i$.

A larger energy deficit $q_i$ implies that less energy should be consumed at location and segment $i$ in order to save more energy for the future so that the energy constraint can be satisfied in the long run. The energy deficit measures the importance of minimizing the energy consumption for the current location and segment. The objective of the task allocation and flying speed control problem is to minimize the delay of task completion time and drone transition time, and hence we combine these two objectives as the weighted sum of the delay and energy consumption at location and segment $i$ as

$$\begin{aligned}\phi_i = \alpha[&\sum_{k=1}^{K_i}\sum_{j=1}^{N}(\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j})x_{ijk} + \frac{L_i}{v_i}] \\ &+ q_i\{\sum_{k=1}^{K_i}\sum_{j=1}^{N}[\frac{(p_i + P_{hov})l_{ik}}{r_{ij}} + \frac{P_{hov}l_{ik}\vartheta_{ik}}{f_j}]x_{ijk} \\ &+ L_i(c_1 v_i^2 + \frac{c_2}{v_i^2})\},\end{aligned} \tag{3.18}$$

where $q_i$ is the weight of the energy consumption and $\alpha$ is the coefficient of the delay to balance the values of the delay and energy consumption. $\phi_i$ is the objective function that we aim to minimize at each location and segment $i$. Note that if the

energy deficit $q_i$ is large, a large weight is put on the energy consumption. It is hence more important to try to consume less energy as possible. Otherwise, the delay should be minimized to optimize the objective function of problem **P0**. Hence, we eliminate the linkage of the energy consumptions of different locations and segments by introducing $p_i$ and incorporating Equation (3.13) into the objective function of problem **P0**. Then, problem **P0** can be transformed into $M$ independent subproblems **P1** at each location and segment. Therefore, problem **P1** should be optimized at each location and segment $i$ as follows,

$$\textbf{P1:} \quad \min_{x_{ijk}} \quad \phi_i$$

$$s.t. \quad (3.12), (3.14), (3.15), (3.16),$$

where $i$ references location and segment $i$. Note that the variables $x_{ijk}$ and $v_i$ in problem **P1** are independent of each other. Hence, problem **P1** can be divided into two subproblems **P1-1** and **P1-2**.

$$\begin{aligned}
\textbf{P1-1:} \quad \min_{x_{ijk}} \quad & \alpha \sum_{k=1}^{K_i} \sum_{j=1}^{N} (\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j}) x_{ijk} \\
& + q_i \sum_{k=1}^{K_i} \sum_{j=1}^{N} [\frac{(p_i + P_{hov}) l_{ik}}{r_{ij}} + \frac{P_{hov} l_{ik} \vartheta_{ik}}{f_j})] x_{ijk} \\
& = \sum_{k=1}^{K_i} \sum_{j=1}^{N} \{ [\alpha + q_i (p_i + P_{hov})] \frac{l_{ik}}{r_{ij}} \\
& \qquad + (\alpha + q_i P_{hov}) \frac{l_{ik} \vartheta_{ik}}{f_j} \} x_{ijk}
\end{aligned}$$

$$s.t. \quad \sum_{j=1}^{N}(\frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j})x_{ijk} \leq D_{ik}, \quad \forall k \in \{1, ..., K_i\}, \tag{3.19}$$

$$\sum_{j=1}^{N} x_{ijk} = 1, \ \forall k \in \{1, ..., K_i\}, \tag{3.20}$$

$$x_{ijk} \in \{0, 1\}, \forall k \in \{1, ..., K_i\}, j \in \mathcal{N}. \tag{3.21}$$

Problem **P1-1** can be considered as the task allocation problem for all tasks at location $i$.

$$\textbf{P1-2:} \quad \min_{v_i} \quad \varphi_i = \alpha\frac{L_i}{v_i} + q_i L_i(c_1 v_i^2 + \frac{c_2}{v_i^2})$$

$$s.t. \quad V_{min} \leq v_i \leq V_{max}. \tag{3.22}$$

Problem **P1-2** can be regarded as the flying speed control problem in segment $i$.

Note that problem **P1-1** is an integer linear programming (ILP) problem and hence is nontrivial to obtain its optimal solutions because of the high computational complexity. We hence design a heuristic approach to solve problem **P1-1**. Each task can only be processed at one fog node (indicated by Equations (3.20) and (3.21)) and the task completion time should not surpass the deadline $D_{ik}$ (indicated by Equation (3.19)). Hence, we define a feasible fog node set $\mathcal{J}_k = \{j \mid \frac{l_{ik}}{r_{ij}} + \frac{l_{ik}\vartheta_{ik}}{f_j} \leq D_{ik}\}$ for each task $k$ that incorporates the indexes of fog nodes that do not violate the task's deadline requirement. The basic idea of the heuristic approach is to choose the fog node (from the feasible fog node set $\mathcal{J}_k$ for each task $k$) which achieves the minimum

objective value $\varrho_{jk} = [\alpha + q_i(p_i + P_{hov})]\frac{l_{ik}}{r_{ij}} + (\alpha + q_i P_{hov})\frac{l_{ik}\vartheta_{ik}}{f_j}$ of problem **P1-1**. Therefore, the solution of problem **P1-1** can be expressed as

$$x_{ijk} = \begin{cases} 1, \text{ if } j = \text{argmin}_j\{\varrho_{jk} \mid j \in \mathcal{J}_k\}, \\ 0, \text{ otherwise,} \end{cases}, \quad \forall k \in \{1, ..., K_i\}. \tag{3.23}$$

Note that problem **P1-2** is a continuous optimization problem. To solve problem **P1-2**, we first calculate the derivative of $\varphi_i$ with respect to $v_i$ (i.e., $\frac{\partial \varphi_i}{\partial v_i}$) and obtain critical points of $v_i$ that correspond to local minima or maxima by letting the derivative be zero. These critical points can be derived from

$$\frac{\partial \varphi_i}{\partial v_i} = -\alpha\frac{L_i}{v_i^2} + q_i L_i(2c_1 v_i - 2\frac{c_2}{v_i^3}) = 0. \tag{3.24}$$

Equation (3.24) can be transformed into

$$2q_i L_i c_1 v_i^4 - \alpha L_i v_i - 2q_i L_i c_2 = 0. \tag{3.25}$$

We denote the set of critical points which satisfy Equation (3.25) as $\mathcal{V}_c$.

**Lemma 4.** *$\varphi_i$ is a convex function with regard to $v_i$ where $v_i > 0$.*

*Proof.* The second derivative of $\varphi_i$ can be calculated as

$$\frac{\partial^2 \varphi_i}{\partial v_i^2} = \frac{\partial}{\partial v_i}(\frac{\partial \varphi_i}{\partial v_i}) = 2\alpha\frac{L_i}{v_i^3} + q_i L_i(2c_i + 6\frac{c_2}{v_i^4}) > 0, \tag{3.26}$$

which completes the proof. □

According to Lemma 4, $\varphi_i$ first monotonically decreases until it reaches the critical point (there is only one critical point) and then monotonically increases within the range $(0, +\infty)$. However, the critical point $V^* = \{v_i \mid v_i \in \mathcal{V}_c, v_i > 0\}$ is not necessarily the optimal $v_i$ because $v_i$ has to fall into the range $[V_{min}, V_{max}]$. Otherwise,

if $V^* < V_{min}$, $\varphi_i$ monotonically increases within $[V_{min}, V_{max}]$. If $V^* > V_{max}$, $\varphi_i$ monotonically decreases. Therefore, the optimal $v_i$ can be calculated as

$$v_i = \begin{cases} V_{min}, \text{ if } V^* < V_{min}, \\ V^*, \text{ if } V_{min} \leq V^* \leq V_{max}, \\ V_{max}, \text{ if } V^* > V_{max}, \end{cases} . \tag{3.27}$$

To sum up, we describe our proposed online algorithm, which provides task allocation and flying speed control strategies at each location and segment, in Algorithm 2. Line 1 initializes the energy deficit. Lines 2-7 determine the task allocation and flying speed for each location and segment in an online fashion. Line 3 calculates the task allocation decisions $x_{ijk}$ for location $i$. Line 4 calculates the flying speed $v_i$ for segment $i$. Line 5 updates the energy deficit $q_{i+1}$. Note that the gap between our proposed online algorithm and the optimal solution is bounded by $\mathcal{O}(\frac{1}{\alpha})$ (demonstrated by Lyapunov optimization technique [52]), where $\alpha$ is the coefficient of delay in Equation (3.18).

**Algorithm 2:** Online Algorithm

**Input :** $M, N, K_i, l_{ik}, \vartheta_{ik}, D_{ik}, f_j, r_{ij}, E, L_i, \alpha, P_{hov}, V_{min}, V_{max}$

**Output:** task allocations $x_{ijk}$, flying speeds $v_i$

Initialize energy deficit $q_1 = 0$ ;

**for** *each location and segment* $i$ **do**

  Calculate task allocation $x_{ijk}$ by solving problem **P1-1** according to Equation (3.23) ;

  Calculate flying speed $v_i$ by solving problem **P1-2** according to Equation (3.27) ;

  Update energy deficit $q_{i+1}$ according to Equation (3.17) ;

  $i = i + 1$ ;

**end**

**return** $x_{ijk}, v_i$;

### 3.4 Performance Evaluation

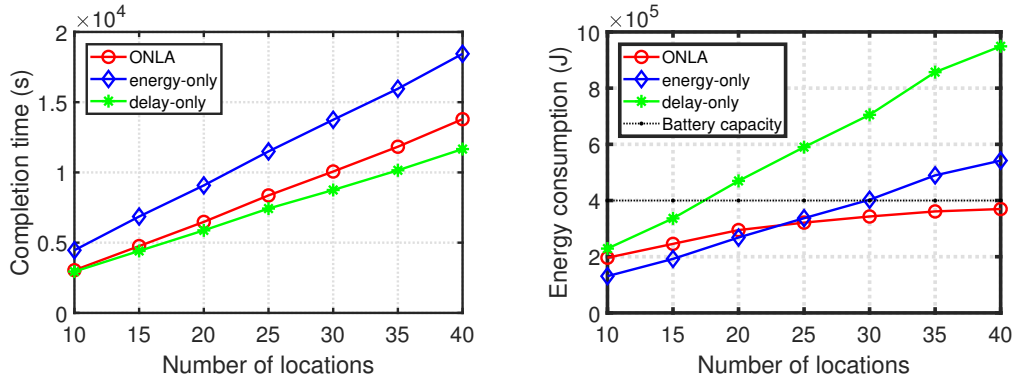
We set up simulations to evaluate the performance of our proposed ONLine Algorithm (ONLA) in this section. We compare ONLA with the comparison algorithm inspired by the existing work in [53]. We denote the comparison algorithm as energy-only, where both the task allocations and flying speeds are chosen with the aim to minimize the drone's energy consumption. Specifically, energy-only assigns each task to the fog node with the minimum energy consumption at each location, and fixes the flying speed at the minimum flying speed. We also utilize the algorithm (denoted as delay-only) inspired by the existing algorithm [54] for comparison, which assigns each task to the fog node with the minimum task completion delay while fixing the flying speed at the maximum flying speed to minimize the transition delay.

In our simulations, we consider a 1800 $m$ $\times$ 1800 $m$ area, where 9 fog nodes are uniformly distributed. There are $M = 30$ locations of interests where the drone

visits, generates computing tasks and offloads them to fog nodes for processing. These locations of interests are uniformly distributed in the area. We assume the transit route of the journey follows the location indexes (i.e., drone flies from location 1 to location 2, and then to location 3,...,$M$, until it reaches the end location 0). The number of tasks generated at each location is randomly distributed from 400 to 600. The task length of each task is randomly distributed from 5 to 10 $Mb$. The computation intensity $\vartheta = 50$ CPU cycles per $Mb$. The fog nodes' computing capacities are randomly chosen from 200 to 800 CPU cycles per second. We assume the height of the flying plane $H = 500$ $m$. The environment-related constants $\omega$ and $\beta$ in Equation (2.1) are 9.6 and 0.28, respectively. The speed of light $c = 3 \times 10^8$ $m/s$. The carrier frequency $f_c$ is 2 $GHz$. The environment-related constants $\xi_{LoS}$ and $\xi_{NLoS}$ in Equations (2.4) and (2.5) are 1 $dB$ and 20 $dB$, respectively [37]. The system bandwidth $W$ is 10 $MHz$. The noise power density $N_0 = -174$ $dBm/Hz$. The wireless transmission power is 3 $W$. The drone mass $m = 500$ $g$ and the earth gravitational acceleration, i.e., $g = 9.8$ $m/s^2$; $r_p, n_p$ and $\rho$ in Equation (2.7) are 20 $cm$, 4, and 1.225 $kg/m^3$, respectively. The propulsion power related parameters are $c_1 = 0.1$ and $c_2 = 200$. The drone's maximum and minimum flying speed are 20 $m/s$ and 3 $m/s$, respectively. Note that the above drone-related parameters are consistent with [46]. All the above parameters are default values and may change if we investigate their impacts on the algorithm performances.

Figure 3.2 compares the performances of the drone's journey completion time and energy consumption among the three algorithms with different numbers of locations ranging from 10 to 40. Figure 3.2(a) and Figure 3.2(b) illustrate the completion time and energy consumption with different numbers of locations, respectively. In Figure 3.2(a), the completion time of the three algorithms increases with the number of locations because more locations bring more task processing time and drone's transition time. Delay-only achieves the smallest completion time and

hence performs the best because delay-only tries to minimize the task completion delay and drone's transition delay without considering the drone's battery capacity. ONLA performs close to delay-only when the number of locations is small, while the difference between them becomes larger when the number of locations increases. Figure 3.2(b) shows that delay-only incurs the most energy consumption and greatly violates the drone's battery constraint when the number of locations is large while the energy consumption of ONLA is always within the battery limit. When the number of locations is small, energy-only performs better than ONLA because it tries to minimize the energy consumption of task allocations. However, when the number of locations is large, the energy consumption of energy-only increases faster than ONLA; it becomes larger than that of ONLA, and even violates the drone's battery constraint because energy-only fixes the flying speed at the minimum flying speed, hence increasing the energy consumption for drone's transitions. On the contrary, ONLA adjusts its flying speed and performs better than energy-only.

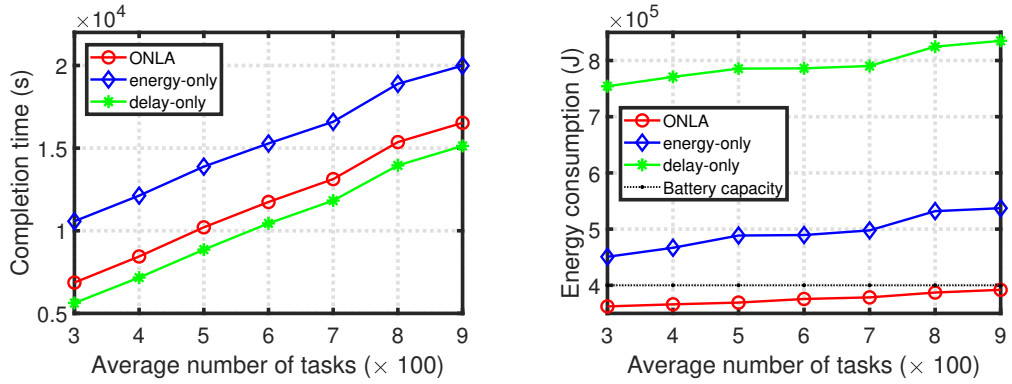



(a) (b)

**Figure 3.2** Completion time and energy consumption vs number of locations.

Figure 3.3 illustrates the journey completion time and energy consumption of the three algorithms with different average numbers of tasks ranging from 300 to 900. We denote the average number of tasks as $\bar{K}$, and then the number of tasks at each location is randomly distributed from $\bar{K} - 100$ to $\bar{K} + 100$. Figure 3.3(a) and Figure

3.3(b) depict the journey completion time and energy consumption with different average numbers of tasks, respectively. In Figure 3.3(a), the journey completion time of all three algorithm increases when the average number of tasks increases because more tasks bring more task processing time. Energy-only incurs the largest completion time while delay-only the least. ONLA performs close to delay-only. However, the energy consumption of delay-only severely surpasses the drone's battery capacity as shown in Figure 3.3(b). In Figure 3.3(b), the energy consumptions of all three algorithms increase when the average number of tasks increases because more tasks incur more energy consumption for wireless data transmissions. The energy consumption of ONLA is always within the battery limit while that of energy-only is more than the limit because more energy-only does not consider the flying control problem and fixes the flying speed at the minimum flying speed.

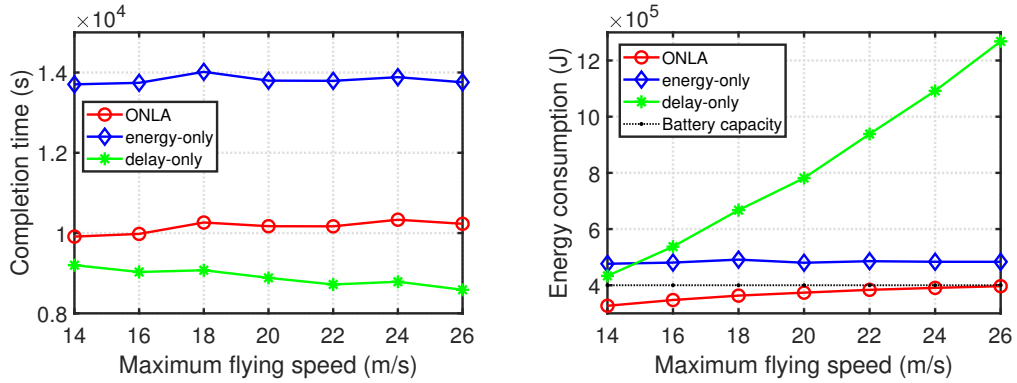



(a) (b)

**Figure 3.3** Completion time and energy consumption vs average number of tasks.

Figure 3.4 evaluates the performance of ONLA with different maximum flying speeds ranging from 14 to 26 *m/s*. The journey completion time and energy consumption with different maximum flying speeds are depicted in Figure 3.4(a) and Figure 3.4(b), respectively. We can observe from Figure 3.4(a) that the journey completion time of delay-only decreases when the maximum flying speed increases while those of ONLA and energy-only do not change much, because delay-only fixes

its flying speed at the maximum flying speed. A larger maximum flying speed reduces the transition time of delay-only and hence the completion time decreases. When the maximum flying speed is small, ONLA performs close to delay-only while their difference becomes large at a high flying speed. In Figure 3.4(b), both delay-only and energy-only exceed the drone's battery limit while ONLA always performs within the limit. The energy consumption of delay-only greatly increases when the maximum flying speed increases because the delay-only tries to minimize the network delay without considering the energy consumption and flies with the maximum flying speed.

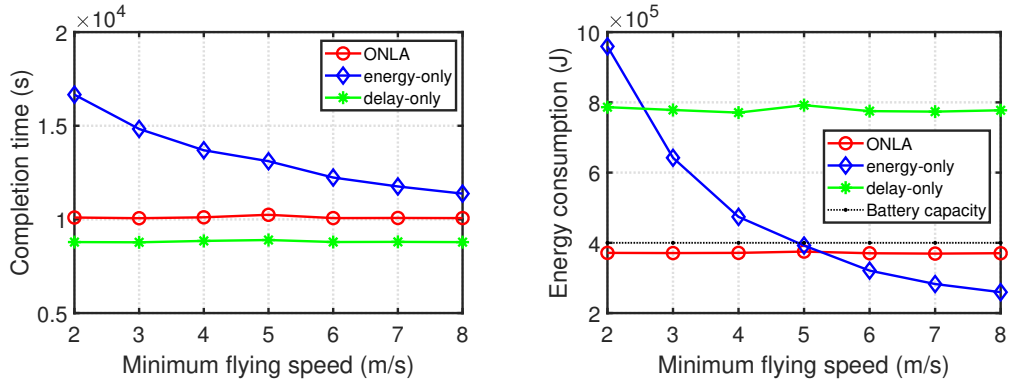



(a) (b)

**Figure 3.4** Completion time and energy consumption vs maximum flying speed.

Figure 3.5 depicts the journey completion time and energy consumption of the three algorithms with different minimum flying speed ranging from 2 to 8 $m/s$. Figure 3.5(a) and Figure 3.5(b) illustrate the completion time and energy consumption with different minimum flying speeds, respectively. In Figure 3.5(a), the journey completion time of ONLA and that of delay-only almost remain the same while that of energy-only greatly decreases with the increase of the minimum flying speed because energy-only fixes its flying speed at the minimum flying speed. A larger minimum flying speed leads to less drone transition time. In Figure 3.5(a), ONLA incurs larger completion time than delay-only, which greatly violates the drone's battery capacity in Figure 3.5(b). We can observe from Figure 3.5(b) that the energy consumption of

energy-only decreases when the minimum flying speed increases. When the minimum flying speed is high, energy-only incurs less energy consumption than that of ONLA, but incurs more journey completion time than that of ONLA as shown in Figure 3.5(a).




(a) (b)

**Figure 3.5** Completion time and energy consumption vs minimum flying speed.

## 3.5 Summary

In this chapter, task allocation and flying control have been jointly optimized in fog-aided IoD networks with the objective to minimize the journey completion time during which all locations of interests are visited and all generated computing tasks are processed. The drone's battery capacity and task completion deadline have been considered as the constraints. This joint optimization problem has been formulated as an MINLP problem. In order to address the challenge of unawareness of future task information, an online algorithm has been proposed. Extensive simulations have demonstrated that the proposed online algorithm performs close to delay-only (which minimizes the journey completion time without considering the drone's battery capacity). Moreover, the proposed algorithm performs better than energy-only (which tries to minimize the drone's energy consumption and maintains a certain flying speed).

# CHAPTER 4

# WIRELESS POWER AND ENERGY HARVESTING CONTROL IN IOD BY DEEP REINFORCEMENT LEARNING

Gharibi *et al.* [30] introduced the conceptual model of an IoD system and detailed its organization, features and implementation. They also explained that the IoD network can be applied for package delivery, disaster rescue, and traffic surveillance. Wazid *et al.* [55] proposed a lightweight user authentication scheme in an IoD network for the users to access data from drones and demonstrated that their scheme provides better security than existing schemes. Bera *et al.* [56] proposed a blockchain based secure framework for data management in IoD networks that provides better security and also incurs less communication and computation overheads. Yao and Ansari [57] designed an online algorithm to address the joint optimization of task allocation and flying speed control in an IoD network to minimize the drone's journey completion time during which a drone generates computing tasks, offloads them to a fog node, and visits different locations of interest.

Energy harvesting is a promising technology to charge batteries. Altinel *et al.* [58] proposed a Markov energy model to analyze the energy outage, shortage and service loss probabilities of an energy harvesting aided communication system. Nguyen *et al.* [59] designed an energy-harvesting-aware routing protocol for IoT networks to improve the lifetime of IoT devices under variable traffic load and energy availability conditions. Yao and Ansari [35] proposed a Stackelberg game in cached-enabled energy-harvesting-aided IoT networks to incentivize the charging station to transmit energy to the IoT devices. Jawad *et al.* [60] utilized the magnetic resonant coupling (MRC) technology for the wireless power transfer to charge the

drone batteries. They demonstrated that the battery life of the drone was extended from 30 to 851 minutes.

The previous work does not consider wireless power control in reducing the energy consumption of IoT/IoD networks. Yao and Ansari [3] jointly optimized the power control and fog resource provisioning in fog-aided IoT networks to minimize the system cost while guaranteeing QoS requirements. Lee and Hong [61] proposed a power control scheme for secure device-to-device communication in IoT networks to improve system energy efficiency. Mach and Becvar [62] proposed a distributed power control algorithm to increase the delivery ratio of computation results constrained by the QoS requirements in mobile edge networks. Yao and Ansari [3] investigated the power control in IoD networks for the data collection service to minimize the drone's energy consumption while satisfying the QoS requirement. However, none of the above works consider the joint optimization of power control and energy harvesting control in IoD networks. Challita *et al.* [63] proposed a deep reinforcement learning algorithm to optimize the transmission power, path, and cell association of each UAV to minimize the interference level and the wireless transmission delay in multi-UAV-aided networks. Pace *et al.* [64] proposed a cognitive transmission power control scheme in IoT networks by a muli-agent Q-learning algorithm where each IoT sensor learns its own power control policy.

Deep reinforcement learning has been utilized in time-varying IoT/IoD networks to improve the performance of network strategies [65]. Lei *et al.* [66] proposed a joint computation offloading and multiuser scheduling problem in IoT edge system to minimize the average weighted sum of delay and power consumption. They further designed a deep reinforcement learning algorithm to solve this joint optimization problem. Yao and Ansari [67] investigated the content placement problem in time-varying cache-enabled IoT networks to minimize the data transmission delay constrained by the cache storage capacity and IoT data freshness. Liu *et al.* [68]

proposed a data collection and secure sharing scheme by combining Ethereum blockchain and deep reinforcement learning to create a reliable and safe IoT environment. However, none of the above works consider utilizing deep reinforcement learning to solve the power control in energy harvesting aided IoD networks.

Our preliminary results of wireless power control in energy harvesting aided IoD networks by deep reinforcement learning was presented at ICC2020 [69]. We extend our preliminary work by additionally considering the energy harvesting control (i.e., determining the amount of transmitted energy to each drone) to further reduce our system energy cost. In this work, we investigate the joint optimization of wireless power control and energy harvesting control in time-varying IoD networks to minimize the long-term average system cost constrained by the drone battery capacities and QoS requirements. A deep reinforcement learning algorithm is proposed to solve this joint optimization problem.

## 4.1 System Model

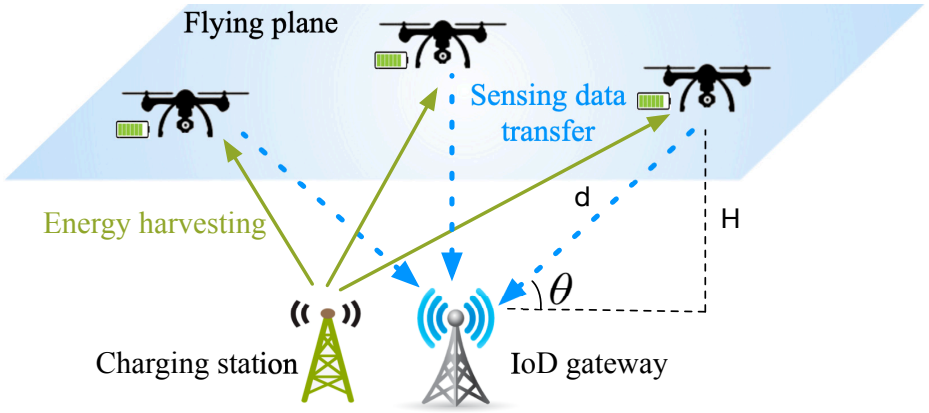



**Figure 4.1** Data collection in energy harvesting aided IoD.

Consider our system model with $N$ drones hovering in the flying plane at the height of $H$, as shown in Figure 5.1. We denote the set of drone indexes as $\mathcal{N} = \{1, 2, ..., N\}$. The drones sense the environmental data (e.g., pictures and videos) at different locations and send them to the IoD gateway. The IoD gateway can further

process the sensed data and send them to a monitor or users who request these data. Owing to the limited drone battery capacities, a charging station is utilized to charge the drone batteries in order to support their operations. Specifically, the drone battery can harvest energy by converting the received radio frequency (RF) signals from the charging station to power [70]. The charging station can use the license-free frequency bands (e.g., 915 *MHz* [71] and 5 *GHz* [72]) for energy transfer and provide controllable energy.

We assume the network operates at discrete time epochs and the network states remain static within a time epoch but vary over different ones [73]. At each time epoch, the IoD gateway determines each drone's wireless transmission power to transmit its sensed data and the transmitted energy from the charging station to each drone. In our work, we characterize the QoS requirement as the minimum data transmission time.

### 4.1.1 Drone Data Transmission Delay

The drone's data transmission rate depends on the wireless channel between the drone and the IoD gateway. The wireless channel gain between drone $i$ and IoD gateway $G_i^{BS}$ is a function of the path loss, i.e.,

$$G_i^{BS} = 10^{-\frac{PL_i}{10}}, \tag{4.1}$$

where $PL_i$ is the path loss between drone $i$ and the IoD gateway according to Equation (2.3). Therefore, drone $i$'s wireless transmission rate $r_i$ can be calculated by the Shannon's formula

$$r_i = W_i \log_2(1 + \frac{p_i G_i^{BS}}{N_0 W_i}), \tag{4.2}$$

where $G_i^{BS}$ is the wireless channel gain between drone $i$ and the IoD gateway, $p_i$ is drone $i$'s wireless transmission power, $W_i$ is the system bandwidth allocated to

drone $i$ and $N_0$ is the noise power spectrum density. Therefore, drone $i$'s wireless transmission time of sending the sensed data to the IoD gateway is

$$\tau_i = \frac{l_i}{r_i} = \frac{l_i}{W_i \log_2(1 + \frac{p_i G_i^{BS}}{N_0 W_i})}, \tag{4.3}$$

where $l_i$ is the data size of drone $i$'s sensed data.

### 4.1.2 Drone's Energy Consumption

Drone $i$'s energy consumption for transmitting the sensing data can be expressed as [41]

$$E_i^{trs} = p_i \tau_i = \frac{p_i l_i}{W_i \log_2(1 + \frac{p_i G_i^{BS}}{N_0 W_i})}, \tag{4.4}$$

where $\tau_i$ is drone $i$'s wireless data transmission time which is defined in Equation (4.3).

We assume RF energy harvesting technology is used to charge the drone batteries, and the amount of the harvested energy depends on the transmitted energy from the charging station and the wireless channel gain between the charging station and the drone. Hence, we utilize the widely used linear energy harvesting model [12] to calculate the harvested energy $E_i^{hrv}$, i.e.,

$$E_i^{hrv} = \eta_i G_i^{EH} e_i, \tag{4.5}$$

where $\eta_i$ is drone $i$'s energy harvesting efficiency, $G_i^{EH}$ is the wireless channel gain between drone $i$ and the charging station and can be similarly calculated by Equation (4.1), and $e_i$ is the transmitted energy from the charging station to drone $i$.

In our work, all drone batteries are rechargeable, and the charged energy can be stored in the battery for future use [74]. We denote the system battery level vector at time epoch $t$ as

$$\boldsymbol{b}(t) = [b_1(t), b_2(t), ..., b_N(t)], \tag{4.6}$$

where $b_i(t) \in [0, B_i^{max}], i \in \mathcal{N}$ is drone $i$'s battery level at time epoch $t$ and is bounded between 0 and the battery capacity $B_i^{max}$. Hence, drone $i$'s battery level evolves from time epoch $t$ to time epoch $t+1$ by

$$b_i(t+1) = \min\{b_i(t) + E_i^{hrv}(t) - E_i^{trs}(t), B_i^{max}\}, \tag{4.7}$$

where $b_i(t+1) \geq 0$, i.e.,

$$b_i(t) + E_i^{hrv}(t) - E_i^{trs}(t) \geq 0, \tag{4.8}$$

which is equivalent to

$$b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1 + \frac{p_i(t) G_i^{BS}}{N_0 W_i})} \geq 0. \tag{4.9}$$

We assume the system energy cost comes from both drone energy consumption and charging station energy consumption, and can be calculated by

$$E_{sys}(t) = c_1 \sum_{i=1}^{N} E_i^{trs}(t) + c_2 \sum_{i=1}^{N} e_i(t), \tag{4.10}$$

where $c_1$ and $c_2$ is the energy cost per joule of drone's battery and charging station, respectively [10]. $e_i(t)$ is the transmitted energy from the charging station to drone $i$ in time epoch $t$.

## 4.2 Problem Formulation

In this section, we formulate the wireless power control and the harvested energy control problem for sensing service in IoD networks, where drones are deployed to sense the environmental information. In order to build an energy efficient system, our objective is to minimize the long-term average system energy cost. Then, the problem can be formulated as

$$\textbf{P0:} \min_{p_i(t),e_i(t)} \frac{1}{M}\sum_{t=1}^{M}[c_1\sum_{i=1}^{N}E_i^{trs}(t)+c_2\sum_{i=1}^{N}e_i(t)] \tag{4.11}$$

$$s.t. \quad p_i(t) \le P_i^m, \ \forall i \in \mathcal{N}, t \in \mathcal{M}, \tag{4.12}$$

$$\frac{l_i}{W_i \log_2(1+\frac{p_i G_i^{BS}}{N_0 W_i})} \le T_i^{th}, \ \forall i \in \mathcal{N}, t \in \mathcal{M}, \tag{4.13}$$

$$b_i(1) = B_i^{max}, \forall i \in \mathcal{N}, \tag{4.14}$$

$$\begin{aligned} b_i(t+1) = \min\{&b_i(t) + \eta_i G_i^{EH} e_i(t) \\ &- \frac{p_i(t) l_i(t)}{W_i \log_2(1+\frac{p_i(t) G_i^{BS}}{N_0 W_i})}, B_i^{max}\}, \ \forall i \in \mathcal{N}, t \in \mathcal{M}, \end{aligned} \tag{4.15}$$

$$\begin{aligned} b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1+\frac{p_i(t) G_i^{BS}}{N_0 W_i})} \ge 0, \\ \forall i \in \mathcal{N}, t \in \mathcal{M}. \end{aligned} \tag{4.16}$$

In Equation (4.11), $M \in \{1, 2, ..., \infty\}$ denotes the total number of time epochs and the objective is to minimize the average system energy cost from time epoch 1 to time epoch $M$. For simplicity, we define $\mathcal{M}$ as the set $\{1, 2, ..., M\}$ to denote time epochs from 1 to $M$. Equation (5.29) imposes drone $i$'s wireless transmission power to be less than the maximum transmission power $P_i^m$. Equation (5.30) is the QoS requirement which imposes drone $i$'s wireless data transmission time to be less than the threshold $T_i^{th}$. Equation (5.31) imposes drone $i$'s initial battery level to be $B_i^{max}$. Equation (4.15) denotes the drone battery level evolution. Equation (4.16)

indicates the feasibility of each drone's battery level. Although the energy consumed for drone's air hovering also accounts for the drone energy consumption [7], it is related to the drone's physical properties (e.g., weight and propellers) and hence is fixed at each equal-length time epoch [39]. The hovering energy consumption is affected by neither the wireless power control nor the energy harvesting control strategies. The energy cost generated by the drone hovering is hence a constant and can be ignored in the objective function which minimizes the average system energy cost. Therefore, we do not include the hovering energy consumption and only focus on the energy consumption for wireless transmission.

**Lemma 5.** *Constraint (4.15) is equivalent to*

$$b_i(t+1) = b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1 + \frac{p_i(t) G_i^{BS}}{N_0 W_i})}, \tag{4.17}$$

*and*

$$\begin{aligned} b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1 + \frac{p_i(t) G_i^{BS}}{N_0 W_i})} \leq B_i^{max}, \\ \forall i \in \mathcal{N}, t \in \mathcal{M}. \end{aligned} \tag{4.18}$$

*Proof.* We use the proof of contradiction to demonstrate this lemma.

Assume that solution $\langle p_i^*(t), e_i^*(t) \rangle$ achieves the minimum system energy cost

$$\phi^* = \frac{1}{M} \sum_{t=1}^{M} [c_1 \sum_{i=1}^{N} E_i^{trs}(t) + c_2 \sum_{i=1}^{N} e_i^*(t)] \tag{4.19}$$

while satisfying that $b_i(t+1) = b_i(t) + \eta_i G_i^{EH} e_i^*(t) - \frac{p_i^*(t) l_i(t)}{W_i \log_2(1 + \frac{p_i^*(t) G_i^{BS}}{N_0 W_i})} > B_i^{max}$. We can always find another $\langle p_i^*(t), \tilde{e}_i(t) \rangle$, where $\tilde{e}_i(t) < e_i^*(t)$, that satisfies $b_i(t+1) = b_i(t) + \eta_i G_i^{EH} \tilde{e}_i(t) - \frac{p_i^*(t) l_i(t)}{W_i \log_2(1 + \frac{p_i^*(t) G_i^{BS}}{N_0 W_i})} \leq B_i^{max}$ and achieves the system energy cost

$$\tilde{\phi} = \frac{1}{M} \sum_{t=1}^{M} [c_1 \sum_{i=1}^{N} E_i^{trs}(t) + c_2 \sum_{i=1}^{N} \tilde{e}_i(t)]. \tag{4.20}$$

Since $\tilde{e}_i(t) < e_i^*(t)$, it can be observed that $\tilde{\phi} < \phi^*$ which violates the assumption that $\phi^*$ is the optimum solution to minimize the system energy cost. Hence, the lemma is proved. □

It is challenging to obtain the global optimum solution of Problem **P0** because of its non-convexity [51]. Additionally, drones' battery levels are coupled with each other over different time epochs and the complete battery level information of all time epochs are required in order to achieve the optimum; this may not be practical in reality. Note that problem **P0** can be considered as a sequential decision-making problem (i.e., wireless transmission power and harvested energy) in a time-varying IoD environment. To solve the time-varying decision-making problem, we first utilize a Markov decision process (MDP) to model the time-varying decision-making problem [13], and then solve the MDP model by a deep reinforcement learning algorithm [75] in the following section.

## 4.3 Algorithm Design

To obtain the solution of problem **P0**, which is a sequential decision-making problem in a time-varying IoD environment, an MDP is utilized to model problem **P0**. We then describe our proposed Power and Energy hArvesting control deep Reinforcement Learning (PEARL) algorithm, which is a modified actor-critic deep reinforcement learning algorithm to solve the MDP model.

### 4.3.1 MDP Model

We use an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{C} \rangle$ to model the power and energy harvesting control in a time-varying IoD network, which consists of the network state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability density functions $\mathcal{F} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, \infty)$, and cost functions $\mathcal{C} : \mathcal{S} \times \mathcal{A} \mapsto [0, \infty)$. Specifically, at each time epoch $t$, the IoD gateway (acting as the network controller) observes the network state $s(t)$ and takes an action

$a(t)$. The network system then generates a system cost $c(t)$ according to the action, and transits to the next network state $s(t+1)$.

**Network State:** We define the network state $s(t)$ at time epoch $t$ as a set of drones' sensed data sizes and battery levels

$$s(t) = [l_1(t), l_2(t), ..., l_N(t), b_1(t), b_2(t), ..., b_N(t)], \tag{4.21}$$

where $l_i(t)$ and $b_i(t)$ are drone $i$'s sensed data size and battery level, respectively. Hence, the network state space $\mathcal{S}$ can be defined as

$$\mathcal{S}(t) = \{s(t) \mid l_i(t) \geq 0, 0 \leq b_i(t) \leq B_i^{max}, i \in \mathcal{N}\}. \tag{4.22}$$

**Action:** The action of the network system $a(t)$ at time epoch $t$ determines $p_i(t)$ (drone $i$'s power control strategy) and $e_i(t)$ (the transmitted energy from the charging station to drone $i$). Hence, $a(t)$ can be defined as

$$a(t) = [p_1(t), p_2(t), ..., p_N(t), e_1(t), e_2(t), ..., e_N(t)]. \tag{4.23}$$

Note that constraints (5.29) and (5.30) must be satisfied, which are equivalent to

$$\frac{N_0 W_i}{G_i^{BS}}(2^{\frac{l_i(t)}{T_i^{th} W_i}} - 1) \leq p_i(t) \leq P_i^m. \tag{4.24}$$

Also, transmitting more energy than the drone battery capacity is a waste of energy in practice, and hence

$$0 \leq e_i \leq B_i^{max}. \tag{4.25}$$

We hence define the action space $\mathcal{A}(t)$ at epoch $t$ as

$$\mathcal{A}(t) = \{\boldsymbol{a}(t) \mid \boldsymbol{a}^{min}(t) \leq \boldsymbol{a}(t) \leq \boldsymbol{a}^{max}(t)\}, \tag{4.26}$$

where

$$\boldsymbol{a}^{min}(t) = [\frac{N_0 W_1}{G_1^{BS}}(2^{\frac{l_1}{T_1^{th} W_1}} - 1), \frac{N_0 W_2}{G_2^{BS}}(2^{\frac{l_2}{T_2^{th} W_2}} - 1), ..., \frac{N_0 W_N}{G_N^{BS}}(2^{\frac{l_N}{T_N^{th} W_N}} - 1), 0, 0, ..., 0], \tag{4.27}$$

and

$$\boldsymbol{a}^{max}(t) = [P_1^m, P_2^m, ..., P_N^m, B^{max}, B^{max}, ..., B^{max}]. \tag{4.28}$$

**System Cost:** The generated cost $c(t)$, defined as the energy cost at time epoch t, is related to the network state $s(t)$ and the taken action $a(t)$. Note that constraints (4.16) and (4.18) should be satisfied, i.e.,

$$0 \leq b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1 + \frac{p_i(t) G_i^{BS}}{N_0 W_i})} \leq B_i^{max}. \tag{4.29}$$

If the taken action $a(t) = [\boldsymbol{p}(t), \boldsymbol{e}(t)]$ violates Equation (4.29), a penalty should be given to the energy cost. Hence, we define the energy cost at time epoch $t$ as

$$c(t) = \begin{cases} \mathbb{M}, \text{ if Equation (4.29) is violated,} \\ \sum_{i=1}^{N} c_1 E_i^{trs}(t) + c_2 e_i(t), \text{ otherwise,} \end{cases} \tag{4.30}$$

where $\mathbb{M}$ is a very large number to penalize the actions that violate Equation (4.29).

$$c(t) = \begin{cases} \sum_{i=1}^{N} c_1 E_i^{trs}(t) + c_2 e_i(t), \\ \qquad\qquad \text{if } 0 \leq b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1 + \frac{p_i(t) G_i^{BS}}{N_0 W_i})} \leq B_i^{max} \\ \mathbb{M}, \text{ otherwise.} \end{cases} \tag{4.31}$$

**Network State Evolution:** The network state $s(t)$ at time epoch $t$ transits to $s(t+1)$ at time epoch $t+1$ according to the taken action $a(t)$. A drone's sensed data size is only related to the dynamic environment and hence drone $i$'s sensed data size $l_i(t)$ at time epoch $t$ and $l_i(t+1)$ at time epoch $t+1$ are independent of each other. On the other hand, the battery levels of different time epochs are coupled with each

other and evolves as $b_i(t+1) = b_i(t) + \eta_i G_i^{EH} e_i(t) - \frac{p_i(t) l_i(t)}{W_i \log_2(1+\frac{p_i(t) G_i^{BS}}{N_0 W_i})}$ (i.e., Equation (4.17)).

**Aim of MDP:** The aim of a general MDP model is to find an action at each time epoch to minimize the accumulated generated cost in the long run [13]. In our system model, the MDP model tries to find the optimal wireless transmission power and the charging station transmitted energy policy $\tau(s, a) = Pr\{a(t) = a \mid s(t) = s\}$, which denotes the probability that action $a$ is taken for a certain state $s$ at time epoch $t$, in order to minimize the accumulated generated cost in the long term. Note that problem **P0** minimizes the long-term average energy cost which is equivalent to minimizing the long-term accumulated energy cost by dividing the total number of time epochs $M$.

To evaluate the long-term generated energy cost, we define the state-action value function [75]

$$Q(s(t), a(t)) = \mathbb{E}\{\sum_{i=t}^{M} \gamma^{(i-t)} c(t)\}, \tag{4.32}$$

which denotes the expected value of all future discounted cost starting from time epoch $t$ in network state $s(t)$ with action $a(t)$ taken. $\gamma \in [0, 1]$ is the discounted factor to measure the importance of future cost. A larger $\gamma$ puts more importance on the future time epochs. For example, the energy costs of future time epochs are of equal importance when $\gamma = 1$. However, we only focus on minimizing the energy cost of time epoch $t$ when $\gamma = 0$. Therefore, the objective of the MDP is to minimize the state-action value function $Q(s(t), a(t))$ starting from the first time epoch 1, i.e.,

$$J(\pi) = \mathbb{E}\{Q(s(0), a(0))\}, \tag{4.33}$$

where $J(\pi)$ is the long-term discounted energy cost.

The basic idea of solving the MDP model is to choose the action with the smallest $Q(\boldsymbol{s}(t), \boldsymbol{a}(t))$ value for network state $s(t)$ at time epoch $t$ [13]. However,

it is challenging to obtain the solution of our MDP model because its actions are continuous. It is impossible to represent all state-action values $Q(s(t), a(t))$. Also, there are infinite action possibilities to be searched and compared in the lookup table where the state-action values are stored [75]. Therefore, to solve the MDP model, we utilize the actor-critic deep reinforcement learning algorithm [14], which is specifically applicable to the time-varying decision making problem with continuous action space.

### 4.3.2 Actor-Critic Deep Reinforcement Learning

The actor-critic deep reinforcement learning learns the optimum action for each time epoch by interacting with the network environment to minimize the generated cost [76]. The basic idea of actor-critic deep reinforcement learning is to combine two deep neural networks (DNNs), i.e., an actor and a critic, to learn optimum power control and energy harvesting control policies. The actor generates continuous actions according to the current network state while the critic evaluates the generated actions and helps the actor update its parameters to generate the actions with better performance, as shown in Figure 4.2.

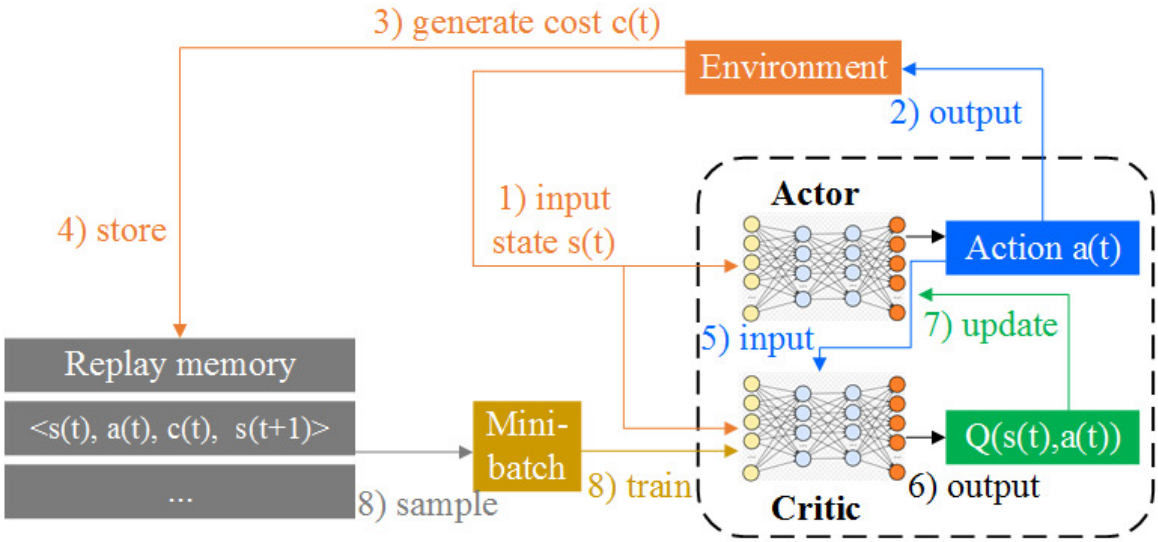



**Figure 4.2** Actor-critic deep reinforcement learning.

**Actor DNN:** The actor uses parameterized function $\pi_\vartheta(s)$, where $\vartheta$ is the parameter of actor DNN, to generate continuous actions for network state $s$. The actor takes the network state $s(t)$ as the input and outputs the action $a(t)$. Specifically, the number of nodes of the actor's input layer is $2N$ which represents the dimension of the network state vector $s(t)$, and the number of nodes of the actor's output layer is $2N$ which represents the dimension of the action $a$. The parameters of the actor DNN is updated by the policy gradient method [14] with the objective to minimize the long-term energy cost $J(\pi_\vartheta)$ (defined in Equation (4.33)). The gradient of the objective function is

$$\nabla_\vartheta J(\pi_\vartheta) = \frac{\partial J(\pi_\vartheta)}{\partial \pi_\vartheta} \frac{\partial \pi_\vartheta}{\partial \vartheta} = \mathbb{E}\{\nabla_a Q_\theta(s,a) \nabla_\vartheta \pi_\vartheta(s)\}, \tag{4.34}$$

where $Q_\theta(s,a)$ is the parametrized station-action value function of the critic and $\theta$ is the critic DNN's parameter. Then, the actor's parameter $\vartheta$ is updated by the gradient descend

$$\vartheta = \vartheta - \omega_a \nabla_\vartheta J(\pi_\vartheta), \tag{4.35}$$

where $\omega_a$ is the actor's learning rate.

Note that the generated action may not be optimal. We hence consider the tradeoff between the exploitation and exploration [75]. Specifically, we prefer to exploit the actions with predicted smallest energy cost (i.e., the generated actions by the actor). Moreover, we still need to explore the unknown actions. Therefore, the chosen action can be calculated as

$$a(t) = \begin{cases} \text{random feasible action, with probability } \epsilon, \\ \text{actor generated } a(t), \text{ with probability } 1-\epsilon, \end{cases} \tag{4.36}$$

where $\epsilon$ is the probability of exploring random actions.

**Critic DNN:** The critic evaluates the actor's generated action by adapting the parametrized state-action value function $Q_\theta(s, a)$, where $\theta$ is the critic DNN's parameter. The critic takes both the network state and the actor's action $[s(t), a(t)]$ as the input, so the node number of the input layer becomes $4N$. The critic outputs the state-action value $Q(s(t), a(t))$ and its node number of the output layer is 1. To improve the accuracy of the critic, its parameter is updated at each time epoch by analyzing the actual generated cost from the environment with the temporal difference method [14]. The temporal difference error $\delta(t)$ is defined to measure the accuracy of the critic, and can be calculated as [14]

$$Q(s(t), a(t)) = c(t) + \gamma Q(s(t+1), a(t+1)), \tag{4.37}$$

$$a(t) = \underset{a}{\operatorname{argmin}}\, Q(s(t), a) \tag{4.38}$$

$$\delta(t) = \ c(t) + \gamma Q_\theta(s(t+1), a(t+1)) - Q_\theta(s(t), a(t)), \tag{4.39}$$

where $s(t)$, $c(t)$, $s(t+1)$, and $a(t+1)$ can be found in the replay memory. The critic's parameter $\theta$ is then updated by the gradient descend to minimize the temporal difference error $\delta(t)$, i.e.,

$$\theta = \theta - \omega_c \nabla_\theta Q_\theta(\boldsymbol{s}(t), \boldsymbol{a}(t)), \tag{4.40}$$

where $\omega_c$ is the critic's learning rate.

**Replay Memory:** To train the critic DNN (i.e., update the critic's parameters), the network state, action, and generated cost should be stored in a replay memory, which is a finite sized first-in-first-out cache. The training sample $\langle s(t), a(t), c(t), s(t+1)\rangle$ is

collected after the action is made by the actor at each time epoch. When the replay memory is full, the old samples will be discarded. At each time epoch, a mini-batch is sampled from the replay memory to train the critic DNN and update its parameter $\theta$.

**Operation Process:** The detailed process of actor-critic deep reinforcement learning is shown in Figure 4.2. At each time epoch, the following steps (i.e., steps 1-8 in Figure 4.2 are processed:

1. The network state $s(t)$ is inputted to the actor DNN.

2. The actor generates the action $a(t)$.

3. The action $a(t)$ acts on the environment and generates the cost $c(t)$.

4. One training sample $\langle s(t), a(t), c(t), s(t+1)\rangle$ is collected and stored in the replay memory.

5. The network state $s(t)$ and the actor's action $a(t)$ are combined and inputted to the critic DNN.

6. The critic generates the state-action value $Q(s(t), a(t))$.

7. The state-action value $Q(s(t), a(t))$ is then utilized to update the actor's parameter $\vartheta$ according to Equations (4.34) and (4.35).

8. A mini-batch is sampled from the replay memory to update the critic's parameter $\theta$ according to Equations (4.39) and (4.40).

### 4.3.3 Modified Actor-Critic Deep Reinforcement Learning

The actor-critic deep reinforcement learning generates the power control and energy harvesting control actions from the action space $\mathcal{A}(t)$ defined in Equation (4.26). The generated action may not be feasible and violate the constraint Equation (4.29). In this case, the actor-critic deep reinforcement learning algorithm adds a penalty (which

is usually a large number) to the generated energy cost. Therefore, it may take a very long time to converge because many infeasible solutions are considered and compared in the action space. In order to address this problem, we propose PEARL which is a modified actor-critic deep reinforcement learning algorithm.

The basic idea of PREAL is to only utilize the power control policy $p_i^*(t), i \in \mathcal{N}$ from the actor-critic deep reinforcement learning algorithm, and substitute $p_i^*(t)$ to the constraint Equation (4.29). Hence, we have

$$\begin{aligned}\frac{1}{\eta_i G_i^{EH}}[\frac{p_i^*(t)l_i(t)}{W_i \log_2(1+\frac{p_i^*(t)G_i^{BS}}{N_0 W_i})} - b_i(t)] &\leq e_i(t) \\ &\leq \frac{1}{\eta_i G_i^{EH}}[B_i^{max} + \frac{p_i^*(t)l_i(t)}{W_i \log_2(1+\frac{p_i^*(t)G_i^{BS}}{N_0 W_i})} - b_i(t)],\end{aligned} \tag{4.41}$$

which is then utilized to constrain the transmitted energy $e_i(t)$. Then, the feasible transmitted energy $e_i^*(t)$ can be calculated by

$$e_i^*(t) = \begin{cases} e_i^{min}(t), \text{ if } e_i(t) < e_i^{min}(t), \\ e_i^{max}(t), \text{ if } e_i(t) > e_i^{max}(t), \quad \forall i \in \mathcal{N}, \\ e_i(t), \text{ otherwise}, \end{cases} \tag{4.42}$$

where we denote

$$e_i^{min}(t) = \frac{1}{\eta_i G_i^{EH}}[\frac{p_i^*(t)l_i(t)}{W_i \log_2(1+\frac{p_i^*(t)G_i^{BS}}{N_0 W_i})} - b_i(t)] \tag{4.43}$$

and

$$e_i^{max}(t) = \frac{1}{\eta_i G_i^{EH}}[B_i^{max} + \frac{p_i^*(t)l_i(t)}{W_i \log_2(1+\frac{p_i^*(t)G_i^{BS}}{N_0 W_i})} - b_i(t)] \tag{4.44}$$

for simplicity. Since the action $[\boldsymbol{p}^*(t), \boldsymbol{e}^*(t)]$ guarantees the feasibility, the generated energy cost from the IoD network can be calculated by

$$c(t) = \sum_{i=1}^{N}[c_1 \frac{p_i^*(t)l_i}{W_i \log_2(1+\frac{p_i^*(t)G_i^{BS}}{N_0 W_i})} + c_2 e_i^*(t)]. \tag{4.45}$$

**Algorithm 3:** PEARL

**Input :** $N, M, G_i^{BS}, G_i^{EH}, N_0, W_i, l_i, T_i^{th}, P_i^m, B_i^{max}$, $c_1, c_2, \gamma, \omega_a, \omega_c, \epsilon$

**Output:** policy $\pi$

Initialize the actor and critic DNNs with weight parameters $\vartheta$ and $\theta$, respectively ;

Initialize the time epoch $t = 1$ ;

Initialize network state $s(1)$ ;

**for** *each time epoch* $t$ **do**

  Calculate the action $a(t) = [\boldsymbol{p}(t), \boldsymbol{e}(t)]$ based on the actor DNN according to Equation (4.36);

  Choose the wireless transmission power $\boldsymbol{p}^*(t) = \boldsymbol{p}(t)$ ;

  Calculate the feasible transmitted energy $\boldsymbol{e}^*(t)$ according to Equation (4.42) ;

  Choose the action $a^*(t) = [\boldsymbol{p}^*(t), \boldsymbol{e}^*(t)]$ ;

  Generate the cost $c(t)$ according to Equation (4.45) ;

  Observe the network state $\boldsymbol{s}(t+1)$ ;

  Store the tuple $\langle s(t), a^*(t), c(t), s(t+1) \rangle$ in the replay memory ;

  Update the actor DNN parameter $\vartheta$ according to Equations (4.34) and (4.35) ;

  Sample a mini-batch of tuples from the replay memory ;

  Update the critic DNN parameter $\theta$ according to Equations (4.39) and (4.40);

  $t \leftarrow t + 1$ ;

**end**

The detailed process of our proposed PEARL is delineated in Algorithm 3. Lines 1-3 initialize the actor, critic, and network state. Lines 4-16 are executed at each time epoch. Line 5 calculates the generated action $a(t)$ based on the actor DNN. Lines 6-8 fix the power control policy $\boldsymbol{p}(t)$, try to find a feasible energy harvesting policy $\boldsymbol{e}^*(t)$, and choose the modified action $a^*(t)$. Lines 9-11 generate the energy cost $c(t)$ and observe the next network state $s(t+1)$. Line 11 stores a training sample $\langle s(t), a^*(t), c(t), s(t+1)\rangle$ in the replay memory. Line 12 updates the actor DNN parameter $\vartheta$. Lines 13-14 sample a mini-batch from the replay memory to update the critic DNN parameter $\theta$.

### 4.4 Performance Evaluation

We setup simulations to evaluate the performances of our proposed algorithm, PEARL, in this section. The simulations are implemented in Python by TensorFlow which is a machine learning platform [77]. The implementation of a real drone testbed will be left as our future work. We compare PEARL with two benchmark algorithms No-energy-control and Greedy. No-energy-control is a deep reinforcement learning algorithm proposed in our ICC2020 paper [69], where only the drones' wireless transmission powers are optimized. Greedy minimizes the transmitted energy at each time epoch to minimize the system energy cost while fixing the wireless transmission power as the maximum power to minimize the wireless transmission delay.

In our simulations, we consider a 1000 $m$ $\times$ 1000 $m$, where the IoD gateway is located at the center of the area. The charging station is located near the IoD gateway. There are $N = 12$ drones deployed in the flying plane at the height of $H = 50$ $m$. The drones are randomly distributed in the flying plane to collect information from the ground. The environment-related parameters in Equation (2.2) are $\alpha = 9.6$ and $\beta = 0.28$. The carrier frequency $f_c = 2$ $GHz$. The speed of light $c = 3 \times 10^8$ $m/s$. The environment-related parameters for calculating the path losses in Equations (2.4) and

(2.5) are $\xi^{LoS} = 1$ and $\xi^{NLoS} = 20$ $dB$. The system bandwidth $W = 20$ $MHz$ and is evenly allocated to all drones. The noise power density $N_0 = -174$ $dBm/Hz$. The amount of sensed data of each drone is randomly chosen from 100 to 200 $Mb$. Each drone's maximum wireless transmission power is $P^m = 5$ $W$. The battery capacity of each drone is $B^{max} = 800$ $J$. The energy harvesting efficiency $\eta = 0.5$. The unit energy cost $c_1$ and $c_2$ are normalized as 1 and $10^{-12}$, respectively. In PEARL, the discounted factor $\gamma = 0.9$, both actor and critic DNN are fully connected and have 1 hidden layer, and each hidden layer has 64 nodes. Note that the above parameters are default values and they may be changed as needed.

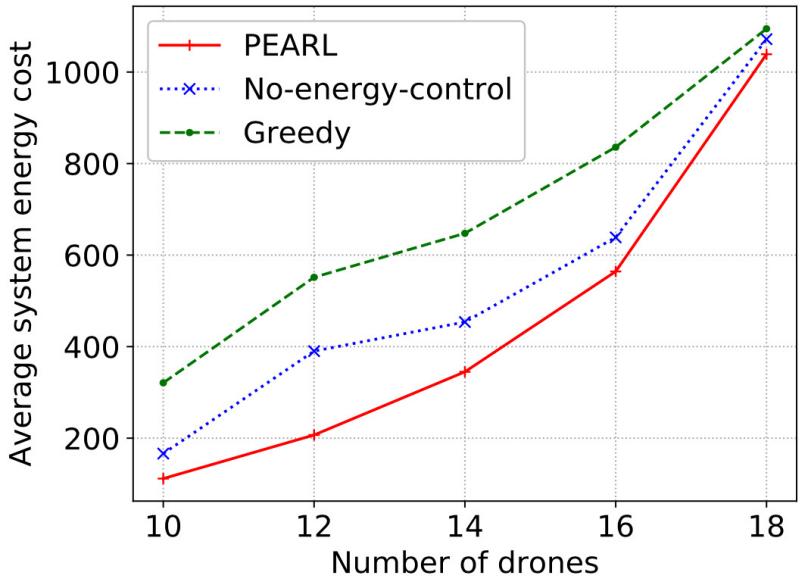



**Figure 4.3** Average system energy cost vs number of drones.

Figure 4.3 illustrates the average system energy cost of three different algorithms after convergence with different numbers of drones ranging from 10 to 18. The average system energy costs of all three algorithms increase with the number of drones because more drones incur more battery charging and a larger amount of transmitted data and hence more energy consumption. PEARL generates the less energy cost than No-energy-control because it jointly optimizes the wireless transmission power and the transmitted energy from the charging station, while No-energy-control only optimizes

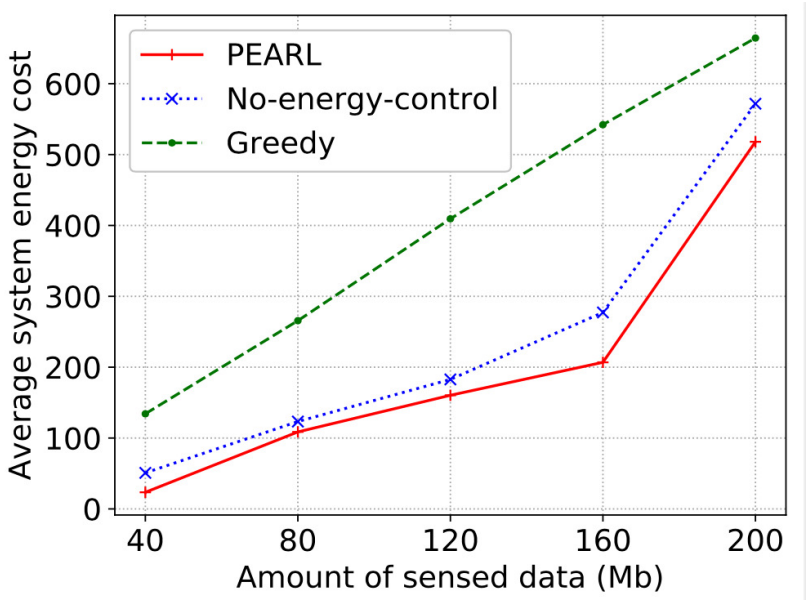



**Figure 4.4** Average system energy cost vs amount of sensed data.

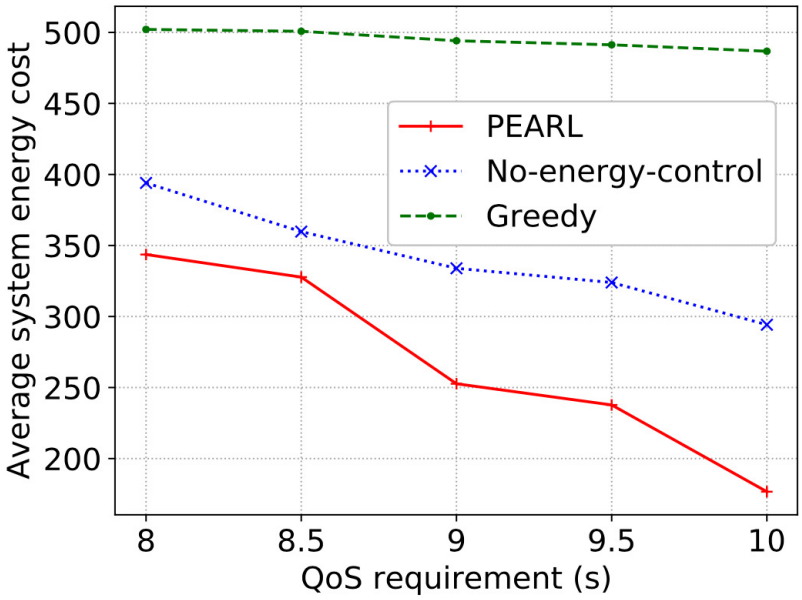



**Figure 4.5** Average system energy cost vs QoS requirement.

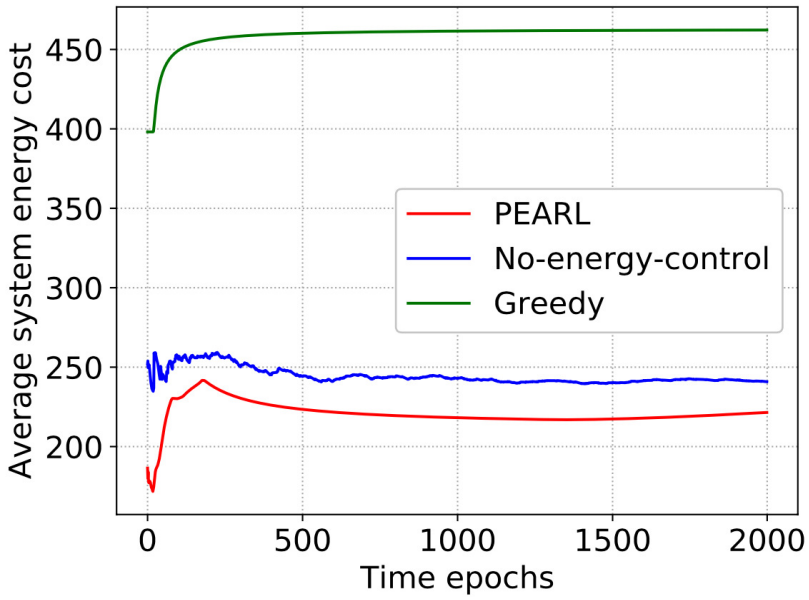



**Figure 4.6** Average system energy cost vs time epochs for different algorithms.

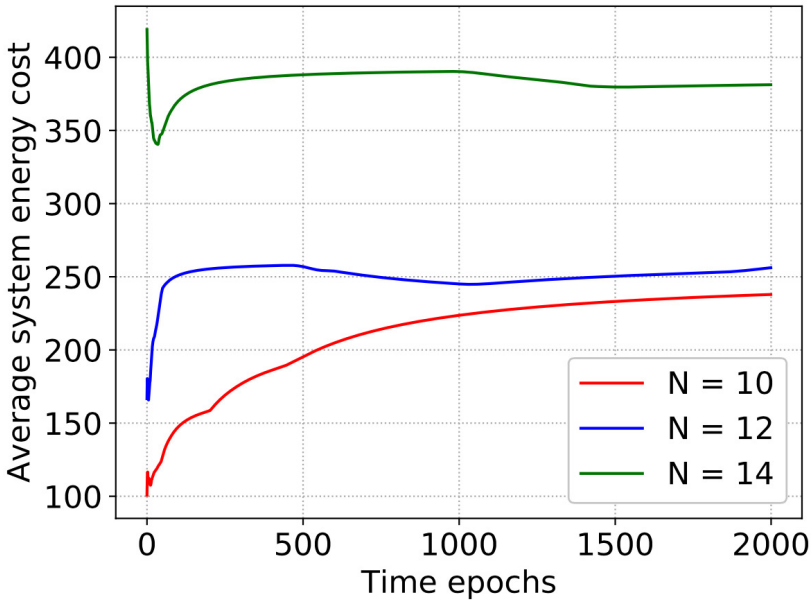



**Figure 4.7** Average system energy cost vs time epochs for different numbers of drones.

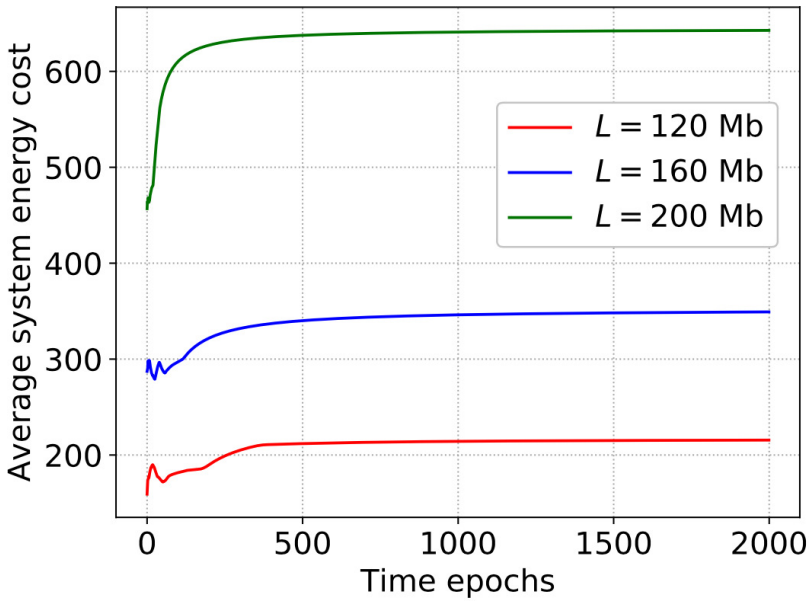



**Figure 4.8** Average system energy cost vs time epochs for different amounts of sensed data.

the transmission power and assumes that all drones' batteries are charged to its fullest. PEARL performs better than Greedy because PEARL considers the policies over different time epochs and utilizes the past experiences to improve its performance, while Greedy only optimizes its solution within one time epoch.

Figure 4.4 compares the average system energy cost of PEARL with that of No-energy-control and Greedy for different amounts of sensed data ranging from 40 to 200 $Mb$. The average system energy costs of all three algorithms become larger when the amount of sensed data increases because more sensed data means more energy is required to transmit these data, thus increasing the system energy cost. PEARL generates the least average system energy cost among the three algorithms for the same reason as in Figure 4.3.

Figure 4.5 evaluates the PEARL's average system energy cost with different QoS requirements (i.e., minimum data transmission delay) ranging from 8 to 10$s$. The average system energy cost of all three algorithms decreases when the QoS requirement becomes less strict (i.e., larger minimum data transmission delay),

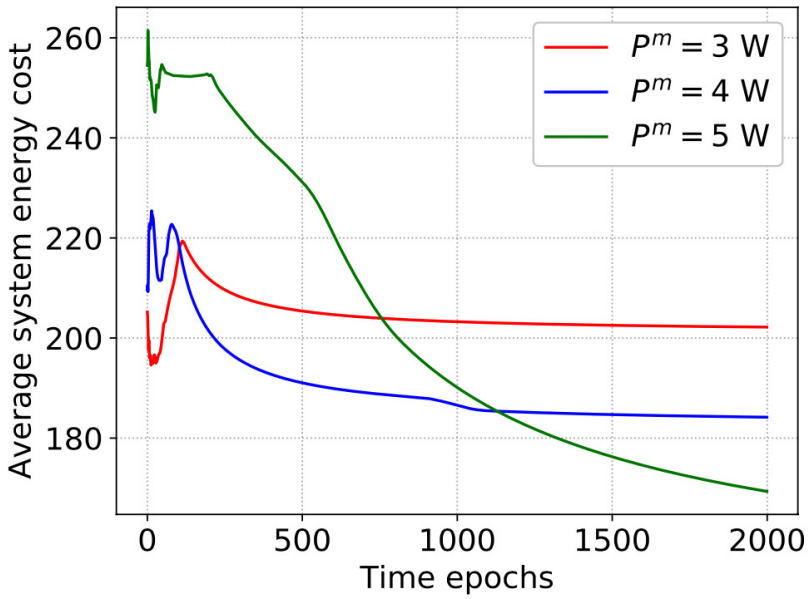



**Figure 4.9** Average system energy cost vs time epochs for different maximum wireless transmission powers.

because a less strict QoS requirement implies that less energy is required to meet the requirement. Similarly, PEARL performs better than No-energy-control and Greedy.

Figure 4.6 illustrates how PEARL, No-energy-control and Greedy converge at different time epochs. Greedy independently optimizes its solutions within each time epoch and so is more likely to obtain similar results at different time epochs when the network status is stable. Hence, Greedy achieves a fast convergence rate. However, both PEARL and No-energy-control are deep reinforcement learning algorithms which are trial-and-error processes, and hence require more time to converge. Additionally, we can observe that PEARL performs the best among the three algorithms for the similar reason in Figure 4.3.

We then investigate the impacts of different parameters on the performance of PEARL in Figs. 4.7 to 4.11. Figure 4.7 illustrates PEARL's average system energy cost for three different numbers of drones including 10, 12, and 14. A larger number of drones incur more energy cost because more drones imply more data to be transmitted and more energy to transmit these data. Figure 4.8 compares the

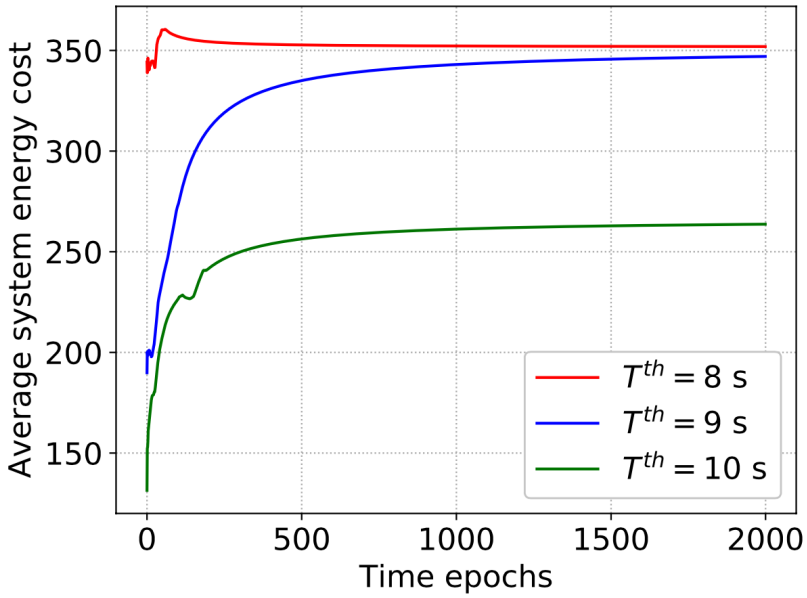



**Figure 4.10** Average system energy cost vs time epochs for different QoS requirements.

PEARL's average system energy costs for different amounts of sensed data including 120, 160, and 200 *Mb*. A larger amount of sensed data incurs more energy cost because more sensed data requires more energy to transmit them. Figure 4.9 evaluates PEARL's average system energy cost for different maximum wireless transmission powers including 3, 4, and 5 *W*. A larger maximum wireless transmission power incurs less energy cost because a larger maximum wireless transmission power provides a larger action space and more possible solutions, hence improving the probability of finding a solution with better performance. However, a larger action space requires more time to converge. Therefore, a larger maximum wireless transmission power incurs a slower convergence rate. Figure 4.10 evaluates PEARL's performance with difference QoS requirements including 8, 9, and 10 *s*. A stricter QoS requirement (i.e., smaller minimum data transmission delay) incurs less system energy cost because less energy is required to meet the QoS requirement. Figure 4.11 illustrates PEARL's average energy cost with different discounted factors including 0.1, 0.5, and 0.9. The discounted factor measures the importance of future time epochs. Since we try to

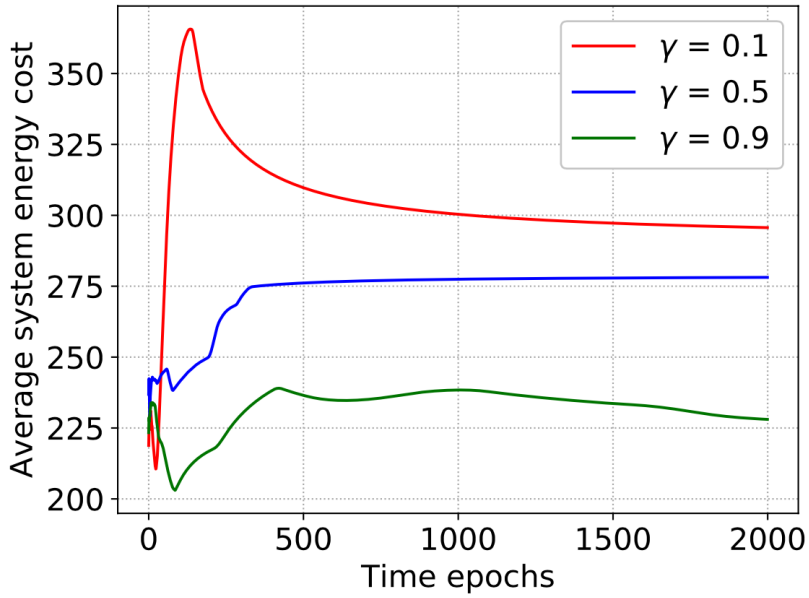



**Figure 4.11** Average system energy cost vs time epochs with different discounted factors.

minimize the long-term average system energy cost, a larger gamma, which puts more importance to future time epochs, achieves a better performance, i.e., less average system energy cost.

## 4.5 Summary

In this chapter, the joint optimization of power control and energy harvesting control has been investigated in time-varying IoD networks. The joint optimization problem has been formulated to determine each drone's wireless transmission power and the transmitted energy from the charging station to each drone at each time epoch with the objective to minimize the long-term average system energy cost constrained by the drones' battery capacities and QoS requirements. An MDP has been formulated to characterize our problem in time-varying IoD networks to show how the network status evolves with different power and energy harvesting control policies. A modified actor-critic deep reinforcement learning algorithm has been designed to solve the problem. Extensive simulations have been conducted to illustrate the impacts of

different parameters on the performance of the proposed algorithm as well as to demonstrate that the our proposed algorithm performs better than the existing algorithms.

# CHAPTER 5

# SECURE FEDERATED LEARNING BY POWER CONTROL IN INTERNET OF DRONES

Fog-aided IoD networks have been investigated to provision services such as object tracking, traffic surveillance, and disaster rescue [3]. Gharibi *et al.* [30] proposed an IoD system to provide navigation services and described how to implement the IoD system. Koubaa and Qureshi [31] proposed a real-time object tracking system where a drone communicates with the network controller to follow a moving object. Hossein *el al.* [78] surveyed various applications, the implementation, and challenges of IoD networks. Wazid *et al.* [55] proposed a user authentication scheme to access the data from drones in IoD networks. Zhou *et al.* [79] jointly optimized the trajectories and transmission power of drones to maximize the secrecy rate. However, none of the above works consider utilizing FL in IoD networks.

Machine learning imparts intelligence into IoT networks by analyzing the data, which are collected by all IoT devices, in the fog node. Meidan *et al.* [80] collected the network traffic data from IoT devices to train a classification model to distinguish the traffic generated by IoT and non-IoT devices. Yao and Ansari [21] constructed a deep reinforcement learning model for the content placement problem in dynamic cache-enabled IoT networks. They also utilized a deep reinforcement learning model to optimize the wireless power control in energy harvesting aided time-varying IoD networks to minimize the average system energy cost [67].

FL has been investigated in wireless networks. Wang *et al.* [23] proposed a control algorithm to minimize the FL loss function constrained by a given resource budget in edge computing systems. Tran *et al.* [81] formulated FL over wireless networks as an optimization problem to balance the tradeoff of the FL learning time,

accuracy level, and energy cost. Wang *et al.* [82] designed an intelligent framework to implement an FL system by utilizing the collaboration among devices and edge nodes and exchanging the learning model parameters. Yang *et al.* [83] formulated an optimization problem to minimize a weighted sum of the FL completion time, local computation energy, and transmission energy for FL in wireless communication networks. However, the above works do not consider the security issue of FL systems. None of the above works exploit the implementation of FL in IoD networks.

The security issue of FL has been studied in several works. Song *et al.* [19] explored the user-level privacy leakage in federated learning and proposed a multi-task generative adversarial network (GAN) framework to identify the anonymized updates of the clients. Lu *et al.* [24] proposed a differentially private asynchronous federated learning scheme for resource sharing in vehicular networks to protect the privacy of updated local models. Xu *et al.* [84] proposed a secure federated training protocol to verify the correctness of results returned from the global aggregator while protecting user data privacy. Wei *et al.* [85] proposed a differential privacy based framework, which adds artificial noise to the uploaded model parameters, to prevent information leakage in federated learning.

Utilizing power control to alleviate the FL's privacy leakage, which is caused by the ground eavesdroppers during the learning parameter uploading in IoD networks, has not been investigated yet. To fill this gap, we optimize the drone wireless transmission powers to maximize the FL system security rate with the consideration of the QoS requirement (i.e., FL training time) and drone battery capacities.

## 5.1 System Model

In a fog-aided IoD network (Figure 5.1), $N$ drones are hovering in the air in the flying plane to collect local data samples and provide the FL service in concert with the fog node to IoD users. We denote $\mathcal{N} = \{1, ..., N\}$ as a set of indexes for indexing drones.

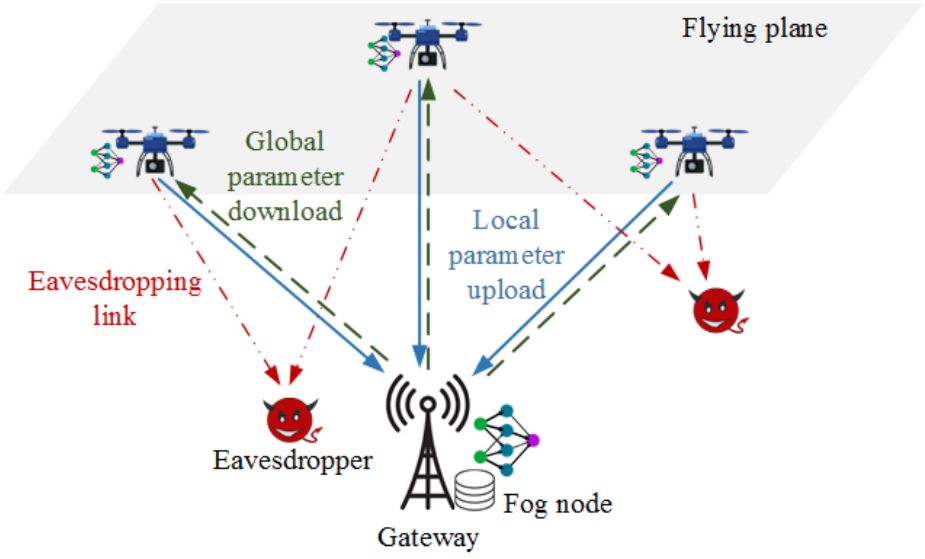



**Figure 5.1** Federated learning in a fog-aided IoD network with eavesdroppers.

The aim of the FL service is to obtain a global machine learning model (e.g., traffic prediction and object recognition). In FL, each drone iteratively downloads the global FL model parameter, updates the parameter with its own local data by local training, and sends it back to the fog node, while the fog node iteratively gathers all updated local parameters and aggregates them to a new global model. The local training is based on the local data samples $\mathcal{D}_n = \{(x_k, y_k)\}$ where $x_k$ is sample $k$'s input (e.g., image pixels) and $y_k$ is the output (e.g., label of the image). A loss function $f_k(w)$ is defined to measure the error of the local model based on data sample $k$, where $w$ is the parameter of the local model. Then, drone $n$'s local training process is to minimize the local loss function [23]

$$F_n(w) = \frac{1}{|\mathcal{D}_n|} \sum_{k \in \mathcal{D}_n} f_k(w), \ \forall n \in \mathcal{N}, \tag{5.1}$$

where $|\mathcal{D}_n|$ is the number of data samples. For simplicity, we define $D_n = |\mathcal{D}_n|$ thereafter. Common examples of loss function $f_k(w)$ include $f_k(w) = \frac{1}{2}\|x_k^T w - y_k\|^2$ for linear regression and $f_k(w) = \{0, 1 - y_k x_k^T w\}, y_k \in \{-1, 1\}$ for support vector machine [86].

Note that $M$ eavesdroppers on the ground aim to steal information from the drones. We denote $\mathcal{M} = \{1, ..., M\}$ as the set of indexes for indexing eavesdroppers. Locations of all eavesdroppers are assumed known, and they can be detected by the leaked power of their radio frequency (RF) front ends [87]. All drones adjust their wireless transmission powers to reduce the possibility of information leakage of the local model parameters.

### 5.1.1 Federated Learning Process

There are global FL iteration and local FL iteration in the FL process (Figure 5.2). In a global iteration, each drone downloads the global parameter from the fog node, trains the model with its local data, and sends the updated local parameter to the fog node. The fog node finally aggregates all updated local parameters into a new global model parameter. The local model parameters are updated by the gradient descent algorithm [88]. In each local iteration, the local parameter is updated according to the gradient of the loss function and learning rate. The relationship between the global iteration and local iteration is shown in Figure 5.2.

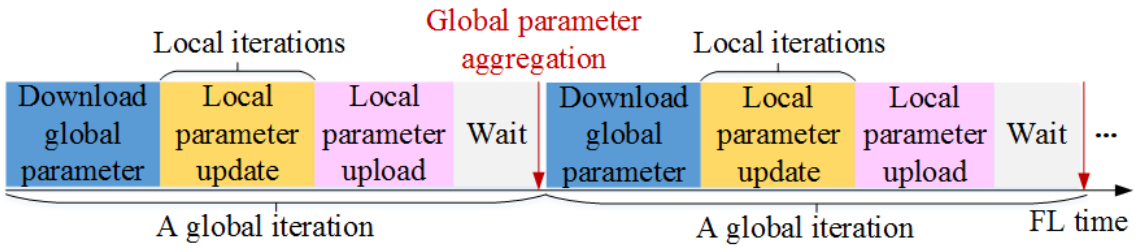



**Figure 5.2** Federated learning process.

The specific FL process is described in Algorithm 4 [88], which is a distributed gradient descent algorithm. The objective of FL is to minimize the global loss function $F(w)$. In the $t$-th global iteration, all the drones first download the global parameter $w^t$ from the fog node, and calculate the gradients of their local loss function $\nabla F_n(w^t)$. Then, the fog node collectes all local gradients and calculates the gradient of the global

**Algorithm 4:** FL Algorithm

Initialize global parameter $w^0$ and global iteration number $t = 0$ ;
**while** *global model accuracy* $\epsilon_g$ *is not obtained* **do**
    Each drone $n$ downloads global parameter $w^t$ from fog node ;
    Each drone $n$ calculates $\nabla F_n(w^t)$ and sends it to fog node ;
    fog node calculates $\nabla F(w^t) = \frac{1}{N} \sum_{n \in \mathcal{N}} \nabla F_n(w^t)$ and broadcasts it to all drones ;
    **for** *each* $n \in \mathcal{N}$ *in parallel* **do**
        Solve local training problem $w_n^{t,*} = \text{argmin}_w \, G_n^t(w)$ by gradient descent algorithm ;
    **end**
    fog node collects all local parameters and calculates $w^{t+1} = \frac{1}{N} \sum_{n \in \mathcal{N}} w_n^{t,*}$ ;
    $t = t + 1$ ;
**end**

loss function

$$\nabla F(w^t) = \frac{1}{N} \sum_{n \in \mathcal{N}} \nabla F_n(w^t), \tag{5.2}$$

which is broadcast to all the drones for local trainings.

Each drone $n$ solves the local training problem

$$\min_{w} \; G_n^t(w) = F_n(w) - [\nabla F_n(w^t) - \eta \nabla F(w^t)]^\top w, \tag{5.3}$$

where $G_n^t(w)$ is the modified loss function of drone $n$ in the $t$-th global iteration, and $\eta$ is a positive constant to control the FL convergence rate [88]. The local training problem is solved by the gradient descent algorithm. We define $w_n^{t,i}$ as drone $n$'s local model parameter at global iteration $t$ and local iteration $i$, and $w_n^{t,*}$ as the local model parameter after convergence, i.e.,

$$w_n^{t,*} = \operatorname*{argmin}_{w} G_n^t(w). \tag{5.4}$$

In the $i$-th local iteration, the local model parameter $w_n^{t,i}$ is updated according to the gradient of $G_n^t(w)$ and the learning rate $\delta$, i.e.,

$$w_n^{t,i+1} = w_n^{t,i} - \delta \nabla G_n^t(w_n^{t,i}), \tag{5.5}$$

where $w_n^{t,0} = w^t$ because it is downloaded from the fog node. According to Equation (5.3), $\nabla G_n^t(w_n^{t,i})$ can be calculated as

$$\nabla G_n^t(w_n^{t,i}) = \nabla F_n(w_n^{t,i}) - \nabla F_n(w^t) + \eta \nabla F(w^t). \tag{5.6}$$

Since $w_n^{t,*}$ is the converged local model parameter, we have

$$\nabla G_n^t(w_n^{t,*}) = \nabla F_n(w_n^{t,*}) - \nabla F_n(w^t) + \eta \nabla F(w^t) = 0, \tag{5.7}$$

The local iteration continues until the local model accuracy $\epsilon_l$ is reached, which is defined as

$$G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*}) \leq \epsilon_l[G_n^t(w^t) - G_n^t(w_n^{t,*})]. \tag{5.8}$$

After all local model parameters $w_n^{t,*}$ are collected, the fog node aggregates all of them into a new global model parameter $w^{t+1}$, i.e.,

$$w^{t+1} = \frac{1}{N}\sum_{n\in\mathcal{N}} w_n^{t,*}, \tag{5.9}$$

The global iteration continues until the global model accuracy $\epsilon_g$ is reached, which is defined as

$$F(w^t) - F(w^*) \leq \epsilon_g[F(w^0) - F(w^*)]. \tag{5.10}$$

### 5.1.2 Federated Learning Convergence Analysis

It is generally impossible to know the exact number of FL iterations, and hence we utilize the convergence bounds to approximate both the local FL iterations and global FL iterations [23]. To analyze the convergence rate of FL, the local loss function $F_n(w)$ of each drone $n$ follows the following assumptions [88]:

- $F_n(w)$ is $\alpha$-strongly convex,
- $F_n(w)$ is $\beta$-smooth.

Assumption 1 implies that [89]

$$\alpha \left\| w - w^{'} \right\| \leq \left\| \nabla F_n(w) - \nabla F_n(w^{'}) \right\|, \ \forall w, w^{'}, \tag{5.11}$$

and

$$F_n(w) - F_n(w^{'}) - \nabla F_n(w)^{\top}(w - w^{'}) \leq -\frac{\alpha}{2}\left\| w - w^{'} \right\|^2, \tag{5.12}$$

where $\|x\|$ denotes the 2-norm of matrix $x$ and $x^{\top}$ is the transpose of matrix $x$.

Assumption 2 implies that [89]

$$\left\|\nabla F_n(w) - \nabla F_n(w^{'})\right\| \leq \beta \left\|w - w^{'}\right\|, \ \forall w, w^{'}, \tag{5.13}$$

and

$$F_n(w) - F_n(w^{'}) - \nabla F_n(w^{'})^{\top}(w - w^{'}) \leq \frac{\beta}{2}\left\|w - w^{'}\right\|^2. \tag{5.14}$$

**Lemma 6.** *$F(w)$ is $\alpha$-strongly convex and $\beta$-smooth.*

*Proof.* According to Equation (5.12), we can derive $\frac{1}{N}\sum_{n\in\mathcal{N}} F_n(w) - \frac{1}{N}\sum_{n\in\mathcal{N}} F_n(w^{'}) - \frac{1}{N}\sum_{n\in\mathcal{N}} \nabla F_n(w)^{\top}(w - w^{'}) \leq \frac{1}{N}\sum_{n\in\mathcal{N}}\{-\frac{\alpha}{2}\left\|w - w^{'}\right\|^2\}$. Combined with Equation (5.2), we have $F(w) - F(w^{'}) - \nabla F(w)^{\top}(w - w^{'}) \leq -\frac{\alpha}{2}\left\|w - w^{'}\right\|^2$, which proves that $F(w)$ is $\alpha$-strongly convex. Similarly, by combining Equation (5.14) and Equation (5.2), we can prove that $F(w)$ is $\beta$-smooth. □

**Lemma 7.** *If both $F(w)$ and $F_n(w)$ are $\alpha$-strongly convex, the following inequations hold:*

$$\left\|\nabla F(w^t)\right\|^2 \geq \alpha[F(w^t) - F(w^*)], \ \forall t, \tag{5.15}$$

*and*

$$\left\|\nabla G_n^t(w_n^{t,i})\right\|^2 \geq \alpha[G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*})], \ \forall i. \tag{5.16}$$

*Proof.* See Appendix B. □

**Lemma 8.** *Local FL problem (5.4) of drone $n$ with the local accuracy $\epsilon_l$ can be solved by the gradient descend method after $I = \frac{2}{(2-\delta\beta)\delta\alpha}\ln(\frac{1}{\epsilon_l})$ iterations, if the local learning rate $\delta < \frac{2}{\beta}$.*

*Proof.* See Appendix C. □

**Lemma 9.** *The global FL algorithm converges after $T = \frac{2\beta^2}{(2\alpha-\beta\eta)\alpha\eta}\ln(\frac{1}{\epsilon_g})$ iterations, if $\eta \in (0, \frac{\alpha}{\beta})$.*

*Proof.* See Appendix D.

□

### 5.1.3 Drone Data Transmission Rate

A drone's data transmission rate depends on the air-to-ground channel between the drone and the fog node. The wireless channel gain between drone $n$ and the fog node is $G_n^D = 10^{-\frac{PL}{10}}$, where $PL$ is defined in Equation (2.3). Therefore, drone $n$'s wireless data transmission rate to the fog node can be calculated as

$$r_n = B_n \log_2(1 + \frac{p_n G_n^D}{N_0 B_n}), \tag{5.17}$$

where $B_n$ is the allocated bandwidth to drone $n$, $p_n$ is drone $n$'s wireless transmission power, and $N_0$ is the noise power spectrum density. Similarly, we can calculate the wireless data transmission rate from drone $n$ to eavesdropper $m$ (i.e., eavesdropping rate):

$$\pi_{n,m} = B_n \log_2(1 + \frac{p_n G_{n,m}^E}{N_0 B_n}), \tag{5.18}$$

where $G_{n,m}^E$ is the wireless channel gain between drone $n$ and eavesdropper $m$.

### 5.1.4 Security Rate

We utilize the security rate to measure the system security level, which is defined as the difference between the drone data transmission rate and the maximum eavesdropping rate [25, 79, 90]. Hence, drone $n$'s security rate is

$$R_n^{SEC} = [r_n - \max_{\forall m \in \mathcal{M}} \pi_{n,m}]^+, \tag{5.19}$$

where $[x]^+ \triangleq \max\{x, 0\}$, $r_n$ is drone $n$'s data transmission rate, and $\pi_{n,m}$ is the eavesdropping rate from drone $n$ to eavesdropper $m$. Note that we intend to maximize the security rates of all drones, and we hence define the system security rate $R^{SEC}$ as

the summation of all drones' security rates, i.e.,

$$R^{SEC} = \sum_{n \in \mathcal{N}} R_n^{SEC} = \sum_{n \in \mathcal{N}} [r_n - \max_{\forall m \in \mathcal{M}} \pi_{n,m}]^+. \tag{5.20}$$

### 5.1.5 Federated Learning Training Time

The FL time in each global iteration consists of both the local computation time for local training and the wireless transmission time to transmit the updated local parameters. Note that we neglect the global parameter download time because it is usually very small. We denote the number of CPU cycles to process one data sample of drone $n$ as $C_n$, which can be measured offline [91]. The number of drone $n$'s data samples is denoted as $D_n$. Hence, the number of CPU cycles for a local iteration is $C_n D_n$. Drone $n$'s local computation time for one local iteration can then be calculated as $\frac{C_n D_n}{f_n}$, where $f_n$ is the CPU computation capacity in CPU cycles per second [91]. Hence, drone $n$'s local computation time is

$$\tau_n^c = I \frac{C_n D_n}{f_n} = \frac{2 \ln(1/\epsilon_l)}{(2 - \delta\beta)\delta\alpha} \frac{C_n D_n}{f_n}. \tag{5.21}$$

Each drone uploads the updated local parameter to the fog node, and the wireless data transmission time for uploading parameters of drone $n$ can be calculated as $\tau_n^w = \frac{s_n}{r_n}$. Note that the global model parameters can only be aggregated until all local model parameters are received in a global iteration. The duration of a global iteration is hence determined by the longest local FL time among all drones. Hence, the FL time of a global iteration can be calculated as

$$\tau^l = \max_{n \in \mathcal{N}} \{\tau_n^c + \tau_n^w\} = \max_{n \in \mathcal{N}} \{\tau_n^c + \frac{s_n}{B_n \log_2(1 + \frac{p_n G_n^D}{N_0 B_n})}\}. \tag{5.22}$$

In summary, the total FL time of all global iterations is

$$\tau = T\tau^l = T \max_{n \in \mathcal{N}} \{\tau_n^c + \frac{s_n}{B_n \log_2(1 + \frac{p_n G_n^D}{N_0 B_n})}\}. \tag{5.23}$$

### 5.1.6 Drone Energy Consumption

The drone's energy is consumed for local model training, wireless data transmission, and hovering in the air.

**Local Computation:** We utilize the widely used energy consumption model which assumes that drone $n$'s energy consumption for processing a single CPU cycle is $\gamma f_n^2$, where $\gamma$ is a constant related to the switched capacitance [92, 93]. Then, drone $n$'s energy consumption for local computation in each global iteration is

$$E_n^c = IC_nD_n\gamma f_n^2 = \frac{2\ln(1/\epsilon_l)}{(2-\delta\beta)\delta\alpha}C_nD_n\gamma f_n^2, \tag{5.24}$$

where $C_nD_n$ is the total number of CPU cycles for one local iteration, and $I$ is the number of local iterations.

**Wireless Data Transmission:** Drone $n$'s energy consumption for uploading the updated local model parameter can be calculated as

$$E_n^w = p_n\tau_n^w = \frac{p_ns_n}{r_n} = \frac{p_ns_n}{B_n\log_2(1+\frac{p_nG_n^D}{N_0B_n})}. \tag{5.25}$$

**Drone Hovering Energy:** The energy consumed for hovering is used for the drone to remain stationary in the air [7]. Drone $n$'s hovering time $\tau^l$ in each global iteration depends on the longest local FL time among all drones. Hence, drone $n$'s hovering energy can be calculated as

$$E_n^{hov} = P^{hov}\tau^l = P^{hov}\max_{n\in\mathcal{N}}\{\tau_n^c + \tau_n^w\}, \tag{5.26}$$

where $P^{hov}$ is the hovering power defined in Equation (2.7).

In summary, the total energy consumption of all drones is

$$E_n = T(E_n^w + E_n^c + E_n^{hov}) = T\frac{p_ns_n}{r_n} + TE_n^c + P^{hov}\tau. \tag{5.27}$$

## 5.2 Problem Formulation

We formulate the power control problem for secure federated learning in a fog-aided IoD network that maximizes the system security rate constrained by the QoS requirement and battery constraint, as problem **P0**.

$$\mathbf{P0:} \quad \max_{\boldsymbol{p}} \sum_{n\in\mathcal{N}} [r_n - \max_{\forall m\in\mathcal{M}} \pi_{n,m}]^+ \tag{5.28}$$

$$s.t. \quad 0 \le p_n \le P_n^m, \ \forall n \in \mathcal{N}, \tag{5.29}$$

$$\tau \le Q^{th}, \tag{5.30}$$

$$E_n \le B_n^{max}, \ \forall n \in \mathcal{N}, \tag{5.31}$$

where $\tau$ and $E_n$ are defined in Equation (5.23) and Equation (5.27), respectively. The objective in Equation (5.28) is to maximize the system security rate. Equation (5.29) imposes the wireless transmission power to be positive and less than the maximum value $P_n^m$. Equation (5.30) is the QoS requirement for the FL service which imposes the FL time not to surpass the requirement $Q^{th}$. Equation (5.31) implies that drone $n$'s energy consumption should be less than its battery capacity $B_n^{max}$. It is challenging to solve problem **P0** because of its non-convexity.

To simplify constraint Equation (5.30), we combine it with Equation (5.22) and (5.23), and we have $T \max_{n\in\mathcal{N}}\{\tau_n^c + \frac{s_n}{B_n \log_2(1+\frac{p_n G_n^D}{N_0 B_n})}\} \le Q^{th}$, which can be transformed into $T(\tau_n^c + \frac{s_n}{B_n \log_2(1+\frac{p_n G_n^D}{N_0 B_n})}) \le Q^{th}, \ \forall n \in \mathcal{N}$. Hence, the lower bound of drone $n$'s wireless transmission power $p_n$ can be calculated as $p_n \ge \frac{N_0 B_n}{G_n^D}[2^{\frac{s_n}{B_n(\frac{Q^{th}}{T}-\tau_n^c)}} - 1]$. We

denote $\tilde{p}_n = \frac{N_0 B_n}{G_n^D}[2^{\frac{s_n}{B_n(\frac{Q_n^{th}}{T}-\tau_n^c)}} - 1]$ for simplicity. Then, $p_n$ satisfies

$$p_n \geq \tilde{p}_n. \tag{5.32}$$

To simplify the objective function Equation (5.28), we have

$$\begin{aligned}
&\sum_{n\in\mathcal{N}}[r_n - \max_{\forall m\in\mathcal{M}} \pi_{n,m}]^+ \\
&= \sum_{n\in\mathcal{N}}[B_n \log_2(1+\frac{p_n G_n^D}{N_0 B_n}) - \max_{\forall m\in\mathcal{M}} B_n \log_2(1+\frac{p_n G_{n,m}^E}{N_0 B_n})]^+ \\
&= \sum_{n\in\mathcal{N}}[B_n \log_2(1+\frac{p_n G_n^D}{N_0 B_n}) - B_n \log_2(1+\frac{p_n \max_{\forall m\in\mathcal{M}} G_{n,m}^E}{N_0 B_n})]^+ \\
&= \sum_{n\in\mathcal{N}}[B_n \log_2(\frac{1+\frac{p_n G_n^D}{N_0 B_n}}{1+\frac{p_n \max_{\forall m\in\mathcal{M}} G_{n,m}^E}{N_0 B_n}})]^+ \\
&= \sum_{n\in\mathcal{N}} B_n \log_2(\frac{1+\frac{p_n G_n^D}{N_0 B_n}}{1+\frac{p_n \max_{\forall m\in\mathcal{M}} G_{n,m}^E}{N_0 B_n}}),\ \text{if } G_n^D \geq \max_{\forall m\in\mathcal{M}} G_{n,m}^E.
\end{aligned} \tag{5.33}$$

We denote $\gamma_n = \frac{G_n^D}{N_0 B_n}$, $\gamma_n' = \frac{\max_{\forall m\in\mathcal{M}} G_{n,m}^E}{N_0 B_n}$, and $\mathcal{N}' = \{n | n \in \mathcal{N}, G_n^D \geq \max_{\forall m\in\mathcal{M}} G_{n,m}^E\}$. Then, the objective becomes

$$\sum_{n\in\mathcal{N}}[r_n - \max_{\forall m\in\mathcal{M}} \pi_{n,m}]^+ = \sum_{n\in\mathcal{N}'} B_n \log_2(\frac{1+\gamma_n p_n}{1+\gamma_n' p_n}) \tag{5.34}$$

Problem **P0** can then be transformed into problem **P1**:

$$\textbf{P1:} \quad \max_{p_n} \quad \sum_{n\in\mathcal{N}'} B_n \log_2(\frac{1+\gamma_n p_n}{1+\gamma_n' p_n}) \tag{5.35}$$

$$s.t. \quad \tilde{p}_n \leq p_n \leq P_n^m,\ \forall n \in \mathcal{N}, \tag{5.36}$$

$$T\frac{p_n s_n}{B_n \log_2(1+\gamma_n p_n)} + TE_n^c + P^{hov}\tau \leq B_n^{max},\ \forall n \in \mathcal{N}, \tag{5.37}$$

$$\tau = T \max_{n \in \mathcal{N}} \{\tau_n^c + \frac{s_n}{B_n \log_2(1 + \gamma_n p_n)}\}, \tag{5.38}$$

where $T$ is the number of FL global iterations defined in Lemma 9, $E_n^c$, defined in Equation (5.24), is drone $n$'s energy consumption for computation in each global iteration, and $\tau_n^c$, defined in Equation (5.21), is drone $n$'s computation time in each FL global iteration. It is still difficult to solve problem **P1** because of its non-convexity. Approaches such as exhaustive search and branch-and-bound are computationally expensive. We hence design an algorithm to tackle this problem with a much lower computational complexity in the next section.

## 5.3 Algorithm Design

We propose the Power Control in Secure FL (PCSF) algorithm in this section to solve problem **P0**. The basic idea of PCSF is to enumerate each possible FL time and optimize all drones' wireless transmission powers, and then choose the best FL time and its corresponding power control policy which achieves the largest system security rate.

We propose the Power Control in Secure FL (PCSF) algorithm in this section to solve problem **P0**. The basic idea of PCSF is to enumerate each possible FL time and optimize all drones' wireless transmission powers, and then choose the best FL time and its corresponding power control policy which achieves the largest system security rate.

### 5.3.1 Subproblem Transformation

Note that the difficulty of problem **P1** lies in the total FL time $\tau$ which couples all $p_n$'s together. In order to solve this challenge, we propose to enumerate each $\tau = T\tau_j^c + \frac{Ts_j}{B_j \log_2(1+\gamma_j p_j)}, \forall j \in \mathcal{N}$ and then compare all derived objective values by different $\tau$. Since $\tau$ is determined, all drones' $p_n$'s are independent. It can be observed that

$B_n \log_2(\frac{1+\gamma_n p_n}{1+\gamma_n^{'} p_n})$ is an increasing function with regard to $p_n$ when $G_n^D \geq \max_{\forall m \in \mathcal{M}} G_{n,m}^E$. Then, maximizing the summation of $B_n \log_2(\frac{1+\gamma_n p_n}{1+\gamma_n^{'} p_n})$ is equivalent to maximizing each $p_n$ when the condition $G_n^D \geq \max_{\forall m \in \mathcal{M}} G_{n,m}^E$ is satisfied. Otherwise, if the drones do not satisfy the condition, their security rates are always zero and do not contribute to the system security rate. To minimize the FL training time, we can choose the maximum wireless transmission power. In summary, all drones try to maximize their wireless transmission power $p_n$ to maximize the system security rate. Therefore, problem **P1** can be transformed into solving $N$ drones' subproblems **P2**:

$$\textbf{P2:} \quad \max_{p_n} \quad p_n \tag{5.39}$$

$$s.t. \quad \tilde{p}_n \leq p_n \leq P_n^m, \tag{5.40}$$

$$T \frac{p_n s_n}{B_n \log_2(1+\gamma_n p_n)} + T E_n^c + P^{hov} \tau_j \leq B_n^{max}, \tag{5.41}$$

$$\tau_j = T \tau_j^c + \frac{T s_j}{B_j \log_2(1+\gamma_j p_j)}, \tag{5.42}$$

where Equation (5.42) means drone $j$ incurs the largest FL time.

### 5.3.2 FL Time Calculation

Since $\tau_j$ is related to variable $p_j$, we first solve the subproblem of drone $j$. Then, $\tau_j$ can be calculated according to $p_j$ and help determine the solutions of other drones' subproblems. By combining constraints Equation (5.41) and Equation (5.42), we have

$$(\frac{B_j^{max}}{T} - E_j^c - P^{hov} \tau_j^c) B_j \log_2(1+\gamma_j p_j) - s_j p_j - P^{hov} s_j \geq 0. \tag{5.43}$$

We define function $g(p_j) = (\frac{B_j^{max}}{T} - E_j^c - P^{hov}\tau_j^c)B_j \log_2(1+\gamma_j p_j) - s_j p_j - P^{hov} s_j \geq 0$. Therefore, the subproblem **P2** of drone $j$ is to find the maximum $p_j$ which satisfies $g(p_j) \geq 0, \tilde{p}_j \leq p_j \leq P_j^m$. We can calculate the derivative of $g(p_j)$ as

$$g^{'}(p_j) = (\frac{B_j^{max}}{T} - E_j^c - P^{hov}\tau_j^c)B_j \frac{\gamma_j \log_2 e}{1+\gamma_j p_j} - s_j. \tag{5.44}$$

Then, we can observe that $g(p_j)$ monotonically increases (i.e., $g^{'}(p_j) < 0$) when $p_j < (\frac{B_j^{max}}{T} - E_j^c - P^{hov}\tau_j^c)B_j \frac{\log_2 e}{s_j} - \frac{1}{\gamma_j}$, and $g(p_j)$ monotonically decreases (i.e., $g^{'}(p_j) > 0$) when $p_j > (\frac{B_j^{max}}{T} - E_j^c - P^{hov}\tau_j^c)B_j \frac{\log_2 e}{s_j} - \frac{1}{\gamma_j}$. Hence, to satisfy $g(p_j) \geq 0$, we have $p_j \in [\lambda, u]$, where $g(\lambda) = 0, g(u) = 0$. Meanwhile, the constraint $\tilde{p}_j \leq p_j \leq P_j^m$ should also be satisfied. We then have $p_j \in [\max\{\lambda, \tilde{p}_j\}, \min\{u, P_j^m\}]$. Therefore, the solution of $p_j$ can be expressed as $p_j = \min\{u, P_j^m\}$. Since $g(p_j)$ decreases when $p_j > (\frac{B_j^{max}}{T} - E_j^c - P^{hov}\tau_j^c)B_j \frac{\log_2 e}{s_j} - \frac{1}{\gamma_j}$, we utilize the binary search method [94] to calculate $u$ which makes $g(u) = 0$.

The basic idea of the binary search method is to repeatedly dividing the search interval in half. Initially, we choose the search interval $[\lambda_1, \lambda_2]$, where $g(\lambda_1) > 0$ and $g(\lambda_2) < 0$. If the value in the middle of the search interval $g(\frac{\lambda_1+\lambda_2}{2}) = 0$, then we find $u = \frac{\lambda_1+\lambda_2}{2}$ and stop the search. Otherwise, if $g(\frac{\lambda_1+\lambda_2}{2}) > 0$, we narrow the search interval to $[\frac{\lambda_1+\lambda_2}{2}, \lambda_2]$ and continue the search. If $g(\frac{\lambda_1+\lambda_2}{2}) < 0$, we narrow the search interval to $[\lambda_1, \frac{\lambda_1+\lambda_2}{2}]$ and continue the search. By the binary search method, we can obtain the value $u$ and $p_j = \min\{u, P_j^m\}$. Based on $p_j$, the FL time $\tau_j = T\tau_j^c + \frac{Ts_j}{B_j \log_2(1+\gamma_j p_j)}$ can be calculated.

### 5.3.3 Subproblem Solution

We then calculate the subproblems of drone $n$ ($n \in \mathcal{N} \setminus j$) based on $\tau_j$. Combining Equations (5.41) and (5.42) yields

$$(B_n^{max} - TE_n^c - P^{hov}\tau_j)B_n \log_2(1+\gamma_n p_n) - Ts_n p_n \geq 0. \tag{5.45}$$

We define function $\xi(p_n) = (B_n^{max} - TE_n^c - P^{hov}\tau_j)B_n \log_2(1+\gamma_n p_n) - Ts_n p_n \geq 0$ and hence $\xi(p_n) \geq 0$. The derivative of $\xi(p_n)$ is

$$\xi^{'}(p_n) = (B_n^{max} - TE_n^c - P^{hov}\tau_j)B_n \frac{\gamma_n \log_2 e}{1+\gamma_n p_n} - Ts_n. \tag{5.46}$$

It can be observed that $\xi(p_n)$ monotonically increases (i.e., $\xi^{'}(p_n) > 0$) when $p_n < (B_n^{max} - TE_n^c - P^{hov}\tau_j)B_n \frac{\log_2 e}{Ts_n} - \frac{1}{r_n}$ and monotonically decreases (i.e., $\xi^{'}(p_n) < 0$) when $p_n > (B_n^{max} - TE_n^c - P^{hov}\tau_j)B_n \frac{\log_2 e}{Ts_n} - \frac{1}{r_n}$. Hence, when $\xi(p_n) \geq 0$, $p_n$ falls within the interval $[\tilde{\lambda}, \tilde{u}]$, where $\xi(\tilde{\lambda}) = 0$ and $\xi(\tilde{u}) = 0$. Note that Equation (5.40) should also be satisfied, and then $p_n \in [\max\{\tilde{\lambda}, \tilde{p}_n\}, \min\{\tilde{u}, P_n^m\}]$.

To calculate $\tilde{u}$, we utilize the binary search method similar to that in Section 5.3.2. Specifically, we first initialize the search interval $[\tilde{\lambda}_1, \tilde{\lambda}_2]$, where $\xi(\tilde{\lambda}_1) > 0$ and $\xi(\tilde{\lambda}_2) < 0$. If $\xi(\frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}) = 0$, we stop the search and assign $\tilde{u} = \frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}$. If $\xi(\frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}) > 0$, we change the search interval to $[\frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}, \tilde{\lambda}_2]$ and continue the search. If $\xi(\frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}) < 0$, we change the search interval to $[\tilde{\lambda}_1, \frac{\tilde{\lambda}_1+\tilde{\lambda}_2}{2}]$ and continue the search. Since we try to maximize $p_n$, we have $p_n = \min\{\tilde{u}, P_n^m\}$.

Note that we assume that $\tau_j = T\max_{n\in\mathcal{N}}\{\tau_n^c + \frac{s_n}{B_n \log_2(1+\gamma_n p_n)}\} = T\tau_j^c + \frac{Ts_j}{B_j \log_2(1+\gamma_j p_j)}$. Hence, we have $\tau_n = T\tau_n^c + \frac{Ts_n}{B_n \log_2(1+\gamma_n p_n)} \leq \tau_j$, which indicates that $p_n \geq \frac{1}{\gamma_n}[2^{\frac{s_n}{B_n(\tau_j/T-\tau_n^c)}} - 1]$. By combining with the QoS requirement Equation (5.32), $p_n$ should satisfy

$$p_n \geq \max\{\frac{1}{\gamma_n}[2^{\frac{s_n}{B_n(\tau_j/T-\tau_n^c)}} - 1], \tilde{p}_n\}, \forall n \in \mathcal{N} \setminus j, \tag{5.47}$$

to denote the candidate condition on checking whether the assumption that drone $j$ has the longest FL training time leads to a feasible solution of problem **P1**.

### 5.3.4 Proposed Algorithm

The basic idea of our proposed algorithm PCSF is to enumerate each possible FL time $\tau_j, \forall j \in \mathcal{N}$ and the corresponding wireless transmission power solutions $p_n$'s.

**Algorithm 5:** PCSF

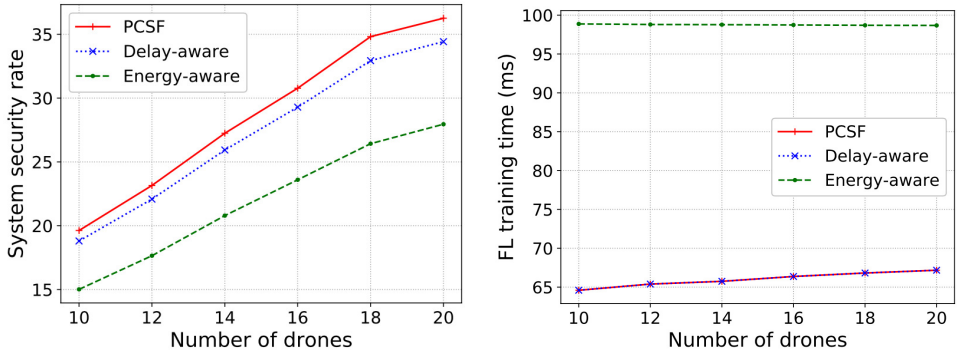
Initialize the candidate vector $V = \emptyset$ ;
**for** *each* $j \in \mathcal{N}$ **do**
    Calculate $p_j$ according to the binary search method in Section 5.3.2 ;
    Calculate FL time $\tau_j = T\tau_j^c + \frac{Ts_j}{B_j \log_2(1+\gamma_j p_j)}$ ;
    **for** *each* $n \in \mathcal{N} \setminus j$ **do**
        Calculate $p_n$ according to the binary search method in Section 5.3.3 ;
    **end**
    **if** *Candidate condition Equation (5.47) is satisfied* **then**
        Calculate the system security rate $R^{SEC}$ ;
        Assign $V[j] = R^{SEC}$ ;
    **else**
        Assign $V[j] = 0$ ;
    **end**
**end**
Choose $j$ that achieves the largest $V[j]$ ;
Choose the FL time $\tau_j$ and its corresponding $p_n$ as the optimum solution.

Then, we choose the best FL time $\tau$, which achieves the largest system security rate, among all possible FL times that satisfy the candidate condition Equation (5.47). The corresponding wireless transmission powers $p_n$ based on the optimum $\tau$ are our final solutions. The detailed process of our proposed algorithm is delineated in Algorithm 5. Lines 2-14 enumerate each possible FL time. Lines 3-4 calculate the FL time $\tau_j$. Lines 5-7 calculate $p_n$'s of all other drones. Lines 8-13 check whether the derived solutions by the current FL time satisfy the candidate condition. Lines 15-16 choose the best solution by comparing all the FL time possibilities. Note that the running time of PCSF is dominated by the binary search in line 6 in the nested loop. The computational complexity of the binary search is $\mathcal{O}(\log_2(\lambda^- - \lambda^+))$, where $[\lambda^+, \lambda^-]$ is the initial interval of the binary search and satisfies $\xi(\lambda^+) > 0$ and $\xi(\lambda^-) < 0$. Therefore, PCSF yields a computational complexity of $\mathcal{O}(N^2 \log_2(\lambda^- - \lambda^+))$.

## 5.4 Performance Evaluation

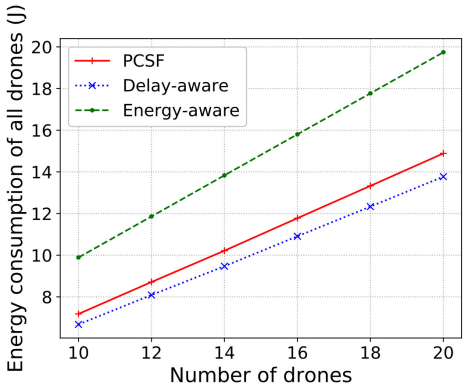
We set up simulations to evaluate the performance of our proposed algorithm PCSF in this section. We compare PCSF with the existing algorithm (denoted as "Delay-aware") inspired by [22] which minimizes the FL training time. We also use the existing algorithm (denoted as "Energy-aware") as the comparison algorithm which is inspired by [21] where the energy consumption for wireless data transmission is minimized.

In our simulations, there are $N = 16$ drones hovering in the flying plane within a 1000 $m \times 1000$ $m$ area to provide the FL service. The drones' locations are randomly distributed in this area and the height of the flying plane is $H = 100$ $m$. The fog node is located in the center of this area to communicate with all drones. There are $M = 3$ eavesdroppers randomly distributed in this area. To calculate wireless channels between drones and the fog node, the environment-related constants $a$ and $b$ are respectively 9.6 and 0.28, the speed of light $c = 3 \times 10^8$ $m/s$, the carrier frequency




(a) System security rate.

(b) FL training time.




(c) Drones' energy consumption.

**Figure 5.3** Key performance metrics vs number of drones.

$f_c = 2\ GHz$, and the environment-related constants $\psi^{LoS} = 1\ dB$ and $\psi^{NLoS} = 20\ dB$. The allocated wireless bandwidth $B = 2\ MHz$ and the noise power density $N_0 = -174\ dBm/Hz$. The above parameters related to drone wireless communications are consistent with [37]. The maximum wireless transmission power $Pm = 3\ W$. To calculate drones' hovering power, each drone's mass $m = 500\ g$ and the earth gravitational acceleration $g = 9.8\ m/s^2$, constants $r_p$, $n_p$ and $\rho$ in Equation (2.7) are 20 $cm$, 4, and 1.225 $kg/m^3$, respectively. The above parameters related to drone hovering power are consistent with [95]. Each drone updates the local model by its local training data, and the number of data samples $D_n$ is randomly chosen from 300 to 500. Each data sample requires $C_n$, randomly chosen from 30 to 50, CPU cycles for computation. The computation capacity of each drone $f = 2 \times 10^9$ CPU cycles per second. The constant $\gamma$ which contributes to the CPU energy consumption of drones is $10^{-28}$ [96]. The battery capacity of each drone $B_n^{max} = 1\ J$. Drone $n$'s uploaded local model parameter $s_n = 5\ Kb$ and the QoS requirement of the FL service $Q^{th} = 200\ ms$. To analyze FL convergence, the loss function is $\alpha = 2$ strongly convex and $\beta = 4$ smooth, constant $\eta = \frac{1}{3}$ in Equation (5.3), and the learning step size of the gradient descent algorithm $\delta = \frac{1}{4}$. The above parameters for FL convergence analysis are consistent with [83]. Note that the above parameters are default values and may change as needed.

We first evaluate PSCF's performances in Figure 5.3 with different numbers of drones ranging from 10 to 20. Figure 5.3(a), Figure 5.3(b), and Figure 5.3(c) depict the performances of the system security rate, FL training time, and all drones' energy consumption, respectively. In Figure 5.3(a), more drones lead to a larger system security rate for all three algorithms because the system security rate is the summation of all drones' security rates. PCSF provides a larger system security rate than those of Delay-aware and Energy-aware. In Figure 5.3(b), the FL training time does not change much when the number of drones increases because all drones'

computations are operated in parallel. PCSF achieves similar FL training time as that of Delay-aware and performs better than Energy-aware. From the objective function of problem **P1**, we can observe that a larger wireless transmission power leads to a larger system security rate. Hence, PCSF prefers larger wireless power to maximize the system security rate. Delay-aware maximizes the transmission power to minimize the FL training time. Therefore, Delay-aware performs close to PCSF as shown in Figure 5.3(a) and Figure 5.3(b). Delay-aware performs better than Energy-aware because Delay-aware prefers higher transmission powers to minimize the FL training time and thus to help improve the system security rate, while Energy-aware prefers lower transmission powers to minimize the energy consumption. In Figure 5.3(c), the energy consumption increases when the number of drones increases because more drones consume more energy. Counterintuitively, Energy-aware incurs the most energy consumption because Energy-aware minimizes the energy consumption for wireless data transmission, while a drone's energy consumption is mostly composed of the hovering energy consumption which is determined by the FL training time. Since Energy-aware incurs the largest FL training time, it incurs the most drone energy consumption. Similarly, Delay-aware achieves the smallest FL training time and hence the least energy consumption. Note that the performance of drones' energy consumption is determined by the FL training time, and we hence only show the performance of FL training time and ignore that of the energy consumption thereafter. In summary, PCSF achieves the largest system security rate and also a small FL training time.

Figure 5.4 illustrates the performances of three algorithms with different numbers of eavesdroppers ranging from 2 to 7. In Figure 5.4, the system security rates of all three algorithms decrease when the number of eavesdroppers increase because more data can be wiretapped. PCSF provides the highest system security

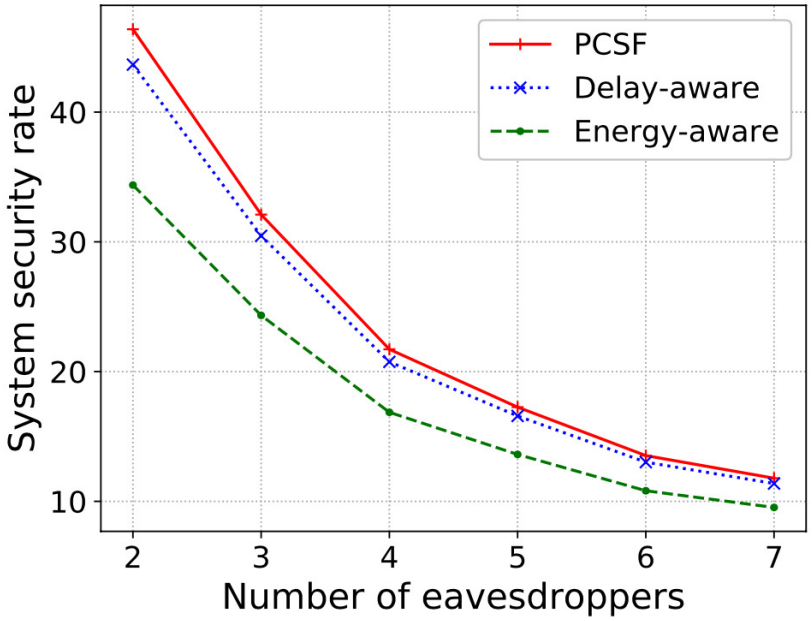
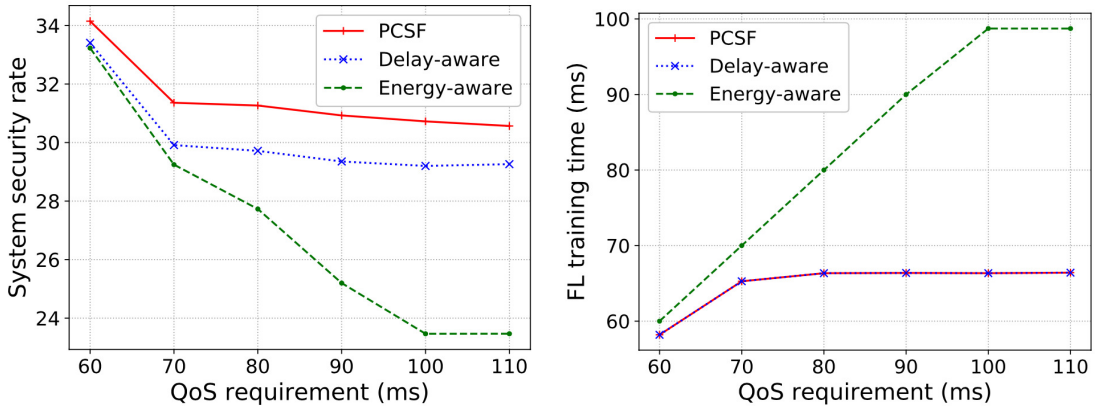



**Figure 5.4** System security rate vs number of eavesdroppers.

rate and Delay-aware the second. Energy-aware achieves the smallest system security rate because of the similar reason as that in Figure 5.3.

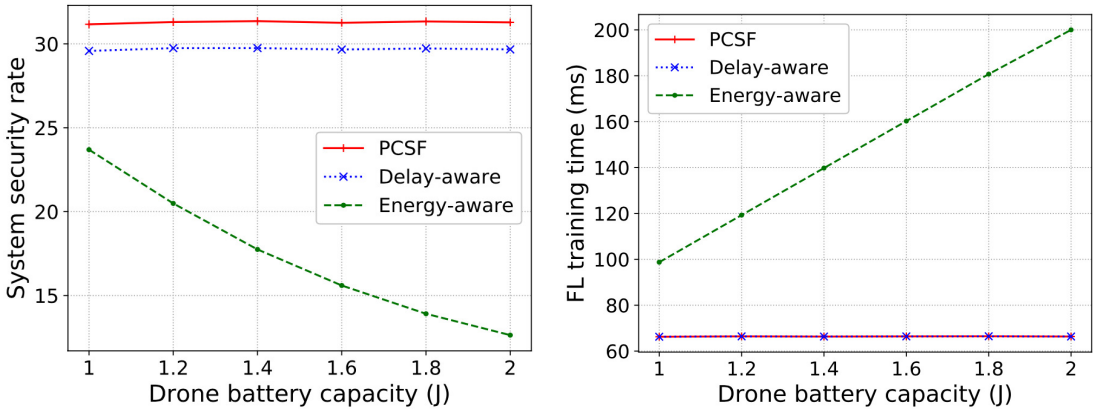
We then investigate the impact of the QoS requirement (i.e., FL training time requirement), ranging from 60 to 110 *ms*, on our proposed algorithm in Figure 5.5. The performances of system security rate and FL training time are shown in Figure 5.5(a) and Figure 5.5(b), respectively. In Figure 5.5(a), the system security rates of all three algorithms decrease with the increase of the QoS requirement. When the QoS requirement is small (i.e., strict), higher transmission powers are required to satisfy the QoS requirement and hence the system security rate is higher. Delay-aware tries to minimize the FL training time and does not increase much when the QoS requirement is larger than 70 *ms* in Figure 5.5(b). Similar to Figure 5.3 and Figure 5.4, PCSF achieves the largest system security rate as shown in Figure 5.5(a) and a small FL training time 5.5(b).

Figure 5.6 evaluates the performances of PCSF with different drone battery capacities ranging from 1 to 2 *J*. The performances of system security rate and FL



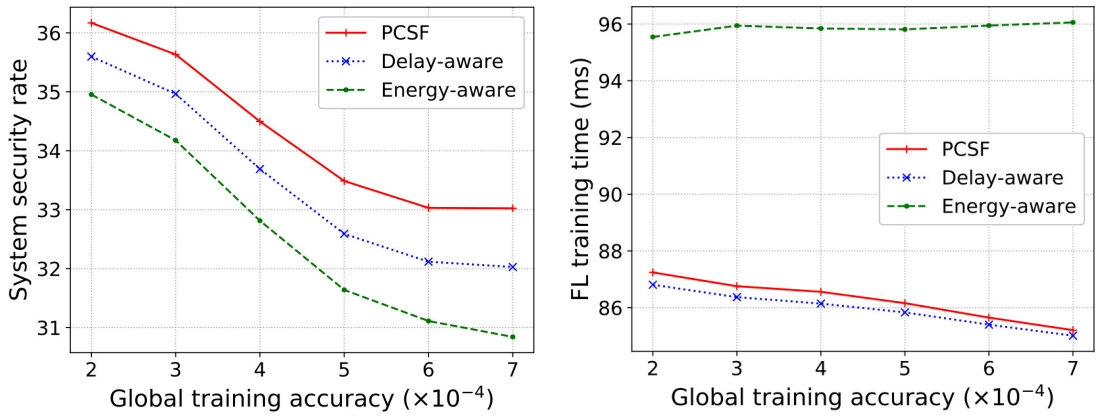

(a) System security rate. (b) FL training time.

**Figure 5.5** Key performance metrics vs QoS requirement.




(a) System security rate. (b) FL training time.

**Figure 5.6** Key performance metrics vs drone battery capacity.




(a) System security rate.

(b) FL training time.

**Figure 5.7** Key performance metrics vs global training accuracy.

training time are shown in Figure 5.6(a) and Figure 5.6(b), respectively. The system security rate in Figure 5.6(a) and the FL training time in Figure 5.6(b) of PCSF and Delay-aware do not change with the increase of the drone battery capacity because the drone battery capacity restricts the minimum wireless transmission power while PCSF and Delay-aware tend to choose the largest transmission power. Energy-aware's system security rate decreases in Figure 5.6(a) and its FL training time increases in Figure 5.6(b) when the drone battery capacity increases, because Energy-aware prefers lower transmission powers which are affected by the increasing drone battery capacity. Moreover, PCSF performs the best among the three algorithms in Figure 5.6(a) and achieves a small FL training time in Figure 5.6(b).

We explore the performances of PCSF with different global training accuracy ranging from $2 \times 10^{-4}$ to $7 \times 10^{-4}$ in Figure 5.7. Figure 5.7(a) and Figure 5.7(b) illustrate the performance of the system security rate and FL training time, respectively. In Figure 5.7(a), the system security rate of all three algorithms decreases when the global training accuracy becomes large. According to Lemma 9, a larger global training accuracy means less global iterations is required and more time and energy consumption are allowed to finish one global iteration, hence reducing

the wireless transmission power to meet the QoS and battery capacity constraints. PCSF provides the largest system security rate among the three algorithms. In Figure 5.7(b), the FL training time of PCSF and Delay-aware decreases with the increase of the global training accuracy while Energy-aware does not change much. Since the transmission power becomes smaller, all three algorithms have larger FL time in one global iteration. Meanwhile, the number of global iterations decreases. Therefore, the FL time in one global iteration increases more than those of PCSF and Delay-aware, and hence the FL training time of Energy-aware remains almost the same while those of PCSF and Delay-aware decrease. We can also observe that PCSF performs close to Delay-aware.

## 5.5 Summary

In this chapter, the secure FL in fog-aided IoD networks has been proposed to counteract eavesdroppers. The FL convergence has been analyzed and demonstrated to calculate the FL training time. The wireless transmission power control problem has been investigated to maximize the system security rate constrained by the QoS requirement and drone battery capacities. The algorithm PCSF has been designed to obtain the solutions of this problem. Simulation results have demonstrated that PCSF performs better than two existing algorithms and achieves both a high system security rate and a small FL training time.

# CHAPTER 6

# CONCLUSION

In this dissertation, the intelligent and secure fog-aided IoD network has been proposed and different issues in the proposed system have been investigated. First, the power control problem in IoD for data collection has been investigated. A sum-of-ratios fractional programming problem has been formulated to minimize the drone's energy consumption constrained by the maximum wireless transmission power and QoS requirement. In order to solve this NP-complete problem, an iteration-based algorithm (PETROL) has been designed, which first obtains the optimal solution of a transformed convex optimization problem by a gradient projection method and then updates the Lagrangian parameters by a modified Newton method. It has been proved that PETROL achieves a linear convergence rate to the optimum solution and a quadratic convergence rate in the neighborhood of the optimum solution. Simulation results have demonstrated PETROL performs better than the existing algorithms.

Then, the task allocation and flying control have been jointly optimized in fog-aided IoD networks with the objective to minimize the journey completion time during which all locations of interests are visited and all generated computing tasks are processed. The drone's battery capacity and task completion deadline are considered as the constraints. This joint optimization problem has been formulated as an MINLP problem. In order to address the challenge of unawareness of future task information, an online algorithm has been proposed. The simulations have demonstrate that our proposed online algorithm performs close to delay-only (which minimizes the journey completion time without considering the drone's battery capacity) and performs better than energy-only (which tries to minimize the drone's energy consumption and maintains a certain flying speed).

After that, the joint optimization of power control and energy harvesting control in time-varying IoD networks has been investigated. The joint optimization problem has been formulated to determine each drone's wireless transmission power and the transmitted energy from the charging station to each drone at each time epoch with the objective to minimize the long-term average system energy cost constrained by the drones' battery capacities and QoS requirements. An MDP has been formulated to characterize our problem in time-varying IoD networks to show how the network status evolves with different power and energy harvesting control policies. A modified actor-critic deep reinforcement learning algorithm has been designed to solve the problem. Extensive simulations have demonstrated that our proposed algorithm performs better than the existing algorithms.

Finally, the secure FL in fog-aided IoD networks has been proposed to counteract eavesdroppers. The FL convergence has been analyzed and demonstrated to calculate the FL training time. The wireless transmission power control problem has been investigated to maximize the system security rate constrained by the QoS requirement and drones battery capacities. This problem has been formulated as a non-linear programming problem to optimize each drone's wireless transmission power. An algorithm PCSF has been designed to obtain the solutions of this problem. Simulation results have demonstrated that PCSF performs better than two existing algorithms and achieves both a high system security rate and a small FL training time.

# APPENDIX A

## PROOF OF LEMMA 3

Note that the convergence rate of PETROL is determined by that of the modified Newton method which determines how $\boldsymbol{x}$ is updated.

Since $p_i(\boldsymbol{x})$ satisfies the Lipschitz condition, $\exists L_1 \ s.t. \ \|p_i(\boldsymbol{x}) - p_i(\bar{\boldsymbol{x}})\| \leq L_1\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$. Since every $f_i(\boldsymbol{x})$ is a continuously differentiable function with regard to $p_i(\boldsymbol{x})$, $f_i(\boldsymbol{x})$ is Lipschitz-continuous [97], i.e., $\exists L_2 \ s.t. \ \|f_i(\boldsymbol{x}) - f_i(\bar{\boldsymbol{x}})\| \leq L_2\|p_i(\boldsymbol{x}) - p_i(\bar{\boldsymbol{x}})\| \leq L_1L_2\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$, where $L_1$ and $L_2$ are Lipschitz constants. Hence, based on Eq. (3.24), $\exists L \ s.t.$

$$\begin{aligned} &\|\boldsymbol{\phi}'(\boldsymbol{x}) - \boldsymbol{\phi}'(\bar{\boldsymbol{x}})\| \\ &= \|diag(f_1(\boldsymbol{x}) - f_1(\bar{\boldsymbol{x}}), \ldots, f_N(\boldsymbol{x}) - f_N(\bar{\boldsymbol{x}}), \\ &\qquad\qquad f_1(\boldsymbol{x}) - f_1(\bar{\boldsymbol{x}}), \ldots, f_N(\boldsymbol{x}) - f_N(\bar{\boldsymbol{x}}))\| \\ &\leq L\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|, \end{aligned} \tag{A.1}$$

where $L = L_1L_2$. We can also derive from Eq. (3.24) that

$$[\boldsymbol{\phi}'(\boldsymbol{x}_k)]^{-1} = diag(\frac{1}{f_1(\boldsymbol{x})}, \ldots, \frac{1}{f_N(\boldsymbol{x})}, \frac{1}{f_1(\boldsymbol{x})}, \ldots, \frac{1}{f_N(\boldsymbol{x})}).$$

Since $p_i(\boldsymbol{x}) > 0$, $f_i(\boldsymbol{x}) = W\log_2(1+r_ip_i(\boldsymbol{x})) \geq W\epsilon$, where $\epsilon$ is a small number. Hence, $\|\frac{1}{f_i(\boldsymbol{x})}\| \leq M$, where $M = \frac{1}{W\epsilon}$. Therefore, $\exists M, s.t.$,

$$\|[\boldsymbol{\phi}'(\boldsymbol{x}_k)]^{-1}\| \leq M. \tag{A.2}$$

For $\delta \in (0,1)$, according to the Newton-Leibnitz formula, Eqs. (3.24) and (2.31), we have

$$\begin{aligned}
&\|\boldsymbol{\phi}(\boldsymbol{x}_k + \delta\boldsymbol{\tau}_k)\| = \|\boldsymbol{\phi}(\boldsymbol{x}_k) + \boldsymbol{\phi}(\boldsymbol{x}_k + \delta\boldsymbol{\tau}_k) - \boldsymbol{\phi}(\boldsymbol{x}_k)\| \\
&= \|\boldsymbol{\phi}(\boldsymbol{x}_k) + \delta \int_0^1 \boldsymbol{\phi}'(\boldsymbol{x}_k + \xi\delta\boldsymbol{\tau}_k)\boldsymbol{\tau}_k d\xi\| \\
&= \|(1-\delta)\boldsymbol{\phi}(\boldsymbol{x}_k) + \delta \int_0^1 [\boldsymbol{\phi}'(\boldsymbol{x}_k + \xi\delta\boldsymbol{\tau}_k) - \boldsymbol{\phi}'(\boldsymbol{x}_k)]\boldsymbol{\tau}_k d\xi\| \\
&\leq (1-\delta)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\| + \delta^2 L\|\boldsymbol{\tau}_k\|^2,
\end{aligned} \tag{A.3}$$

where we utilize $\|\boldsymbol{\phi}'(\boldsymbol{x}_k + \xi\delta\boldsymbol{\tau}_k) - \boldsymbol{\phi}'(\boldsymbol{x}_k)\| \leq L\xi\delta\|\boldsymbol{\tau}_k\|$ derived from Eq. (A.1).

From Eqs. (2.31) and (A.2), we have

$$\|\boldsymbol{\tau}_k\| = \|[\boldsymbol{\phi}'(\boldsymbol{x}_k)]^{-1}\boldsymbol{\phi}(\boldsymbol{x}_k)\| \leq M\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|. \tag{A.4}$$

Hence, Eq. (A.3) can be transformed into

$$\begin{aligned}
\|\boldsymbol{\phi}(\boldsymbol{x}_k + \delta\boldsymbol{\tau}_k)\| &\leq (1-\delta)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\| + \delta^2 L\|\boldsymbol{\tau}_k\|^2 \\
&\leq (1-\delta)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\| + \delta^2 LM^2\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|^2 \\
&= [1 - \delta(1 - \delta LM^2\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|)]\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|.
\end{aligned} \tag{A.5}$$

Let $1-\bar{\delta}_k LM^2\|\boldsymbol{\phi}(\boldsymbol{x}_k)\| = \epsilon$, i.e., $\bar{\delta}_k = \frac{1-\epsilon}{LM^2\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|}$. Then, Eq. (A.5) can be transformed to

$$\|\boldsymbol{\phi}(\boldsymbol{x}_k + \delta_k\boldsymbol{\tau}_k)\| \leq (1 - \epsilon\delta_k)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|, \tag{A.6}$$

where $\delta_k = \min\{1, \bar{\delta}_k\}$.

From Eq. (A.6), we can observe that $\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|$ decreases when $k$ increases. Hence, $\bar{\delta}_k$ is an increasing sequence. Therefore, $\delta_k$ increases until it reaches 1; then it remains the same when $k$ increases. We denote $\bar{k}$ as the particular number of iterations that makes $\delta_{\bar{k}} = 1$.

Case 1: $k \leq \bar{k}$, where $\delta_k$ is an increasing sequence. Hence, $1-\epsilon\delta_k \leq 1-\epsilon\delta_{k-1} \leq \cdots \leq 1-\epsilon\delta_1$. From Eq. (A.6), we can conclude

$$\begin{aligned} \|\boldsymbol{\phi}(\boldsymbol{x}_{k+1})\| &\leq (1-\epsilon\delta_1)\|\boldsymbol{\phi}(\boldsymbol{x}_k)\| \leq (1-\epsilon\delta_1)^2\|\boldsymbol{\phi}(\boldsymbol{x}_{k-1})\| \\ &\leq \cdots \leq (1-\epsilon\delta_1)^k\|\boldsymbol{\phi}(\boldsymbol{x}_1)\|, \end{aligned} \tag{A.7}$$

where $1-\epsilon\delta_1$ is a constant. Therefore, $\|\boldsymbol{\phi}(\boldsymbol{x}_k)\|$ achieves a linear convergence rate.

Case 2: $k > \bar{k}$, where $\delta_k = 1$. Eq. (2.30), i.e., the iteration equation, becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [\boldsymbol{\phi}'(\boldsymbol{x}_k)]^{-1}\boldsymbol{\phi}(\boldsymbol{x}_k). \tag{A.8}$$

Then, it becomes the Newton method which achieves a quadratic convergence rate in the neighborhood of the optimum solution [98]. Lemma 3 is thus proved by combining Case 1 and Case 2.

# APPENDIX B
# PROOF OF LEMMA 7

Since $w^*$ and $w_n^{t,*}$ are the optimal solution of $F(w)$ and $G_n^t(w)$ respectively, we can derive that $\nabla F(w^*) = 0$ and $\nabla G_n^t(w_n^{t,*}) = 0$. We then have

$$\begin{aligned}
\left\|\nabla F(w^t)\right\|^2 &= \left\|\nabla F(w^t) - \nabla F(w^*)\right\|^2 \\
&\overset{(5.11)}{\geq} \alpha \left\|\nabla F(w^t) - \nabla F(w^*)\right\| \left\|w^t - w^*\right\|^2 \\
&= \alpha \nabla F(w^t)^\top (w^t - w^*) \\
&\overset{(5.12)}{\geq} \alpha[F(w^t) - F(w^*) + \frac{\alpha}{2}\left\|w^t - w^*\right\|^2] \\
&\geq \alpha[F(w^t) - F(w^*)],
\end{aligned} \tag{B.1}$$

which proves Eq. (5.15). Meanwhile, we have

$$\begin{aligned}
&\left\|\nabla G_n^t(w_n^{t,i})\right\|^2 = \left\|\nabla G_n^t(w_n^{t,i}) - \nabla G_n^t(w_n^{t,*})\right\|^2 \\
&\overset{(5.7)}{=} \left\|\nabla G_n^t(w_n^{t,i})\right\| \left\|\nabla F_n(w_n^{t,i}) - \nabla F_n(w_n^{t,*})\right\| \\
&\overset{(5.11)}{\geq} \alpha \left\|\nabla G_n^t(w_n^{t,i})\right\| \left\|w_n^{t,i} - w_n^{t,*}\right\| \\
&\overset{(5.6)}{=} \alpha[\nabla F_n(w_n^{t,i}) - \nabla F_n(w^t) + \eta \nabla F(w^t)]^\top (w_n^{t,i} - w_n^{t,*}) \\
&= \alpha\{\nabla F_n(w_n^{t,i})^\top (w_n^{t,i} - w_n^{t,*}) \\
&\qquad - [\nabla F_n(w^t) - \eta \nabla F(w^t)]^\top (w_n^{t,i} - w_n^{t,*})\} \\
&\overset{(5.12)}{\geq} \alpha\{F_n(w_n^{t,i}) - F_n(w_n^{t,*}) \\
&\qquad - [\nabla F_n(w^t) - \eta \nabla F(w^t)]^\top (w_n^{t,i} - w_n^{t,*})\} \\
&\overset{(5.6)}{=} \alpha[G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*})],
\end{aligned} \tag{B.2}$$

which proves Eq. (5.16).

# APPENDIX C
# PROOF OF LEMMA 8

We demonstrate the relationship between $G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*})$ and $G_n^t(w^t) - G_n^t(w_n^{t,*})$. The demonstration process is inspired by [83].

$$\begin{aligned}
G_n^t(w_n^{t,i+1}) &\overset{(5.3)}{=} F_n(w_n^{t,i+1}) - [\nabla F_n(w^t) - \eta \nabla F(w^t)]^\top w_n^{t,i+1} \\
&\overset{(5.5),(5.14)}{\geq} F_n(w_n^{t,i}) - \delta \nabla F_n(w_n^{t,i})^\top \nabla G_n^t(w_n^{t,i}) \\
&+ \frac{\delta^2 \beta}{2} \left\| \nabla G_n^t(w_n^{t,i}) \right\|^2 + [\nabla F_n(w^t) - \eta \nabla F(w^t)]^\top w_n^{t,i+1} \\
&\overset{(5.3),(5.5)}{=} G_n^t(w_n^{t,i}) - \delta G_n^t(w_n^{t,i})^\top G_n^t(w_n^{t,i}) + \frac{\delta^2 \beta}{2} \left\| \nabla G_n^t(w_n^{t,i}) \right\|^2 \\
&= G_n^t(w_n^{t,i}) - \frac{(2 - \delta\beta)\delta}{2} \left\| \nabla G_n^t(w_n^{t,i}) \right\|^2 \\
&\overset{(5.16)}{\leq} G_n^t(w_n^{t,i}) - \frac{(2 - \delta\beta)\delta\alpha}{2} [G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*})].
\end{aligned} \tag{C.1}$$

Based on the above analysis, we have

$$\begin{aligned}
&G_n^t(w_n^{t,i}) - G_n^t(w_n^{t,*}) \\
&\leq [1 - \frac{(2 - \delta\beta)\delta\alpha}{2}][G_n^t(w_n^{t,i-1}) - G_n^t(w_n^{t,*})] \leq ... \leq \\
&\leq [1 - \frac{(2 - \delta\beta)\delta\alpha}{2}]^i [G_n^t(w^t) - G_n^t(w_n^{t,*})] \\
&\leq e^{-i\frac{(2-\delta\beta)\delta\alpha}{2}} [G_n^t(w^t) - G_n^t(w_n^{t,*})],
\end{aligned} \tag{C.2}$$

where the last inequality holds because $1 - x \leq e^{-x}, x \geq 0$. If we assign the local accuracy $e_l = e^{-i\frac{(2-\delta\beta)\delta\alpha}{2}}$, i.e., $I = \frac{2}{(2-\delta\beta)\delta\alpha} \ln(\frac{1}{\epsilon_l})$, the local convergence definition (Eq. (5.8)) holds. Therefore, Lemma 8 is proved.

## APPENDIX D
## PROOF OF LEMMA 9

We demonstrate the relationship between $F(w^t) - F(w^*)$ and $F(w^0) - F(w^*)$.

$$
\begin{aligned}
F(w^{t+1}) &\overset{(5.9)}{=} F(w^t + \frac{1}{N}\sum_{n\in\mathcal{N}}(w_n^{t,*} - w^t)) \\
&\overset{\text{Lemma 6}}{\leq} F(w^t) + \frac{1}{N}\sum_{n\in\mathcal{N}}\nabla F(w^t)^\top (w_n^{t,*} - w^t) \\
&\quad + \frac{\beta}{2}\left\|\frac{1}{N}\sum_{n\in\mathcal{N}}(w_n^{t,*} - w^t)\right\|^2 \\
&\overset{(5.3)}{=} F(w^t) + \frac{1}{N\eta}\sum_{n\in\mathcal{N}}[G_n^t(w_n^{t,*}) - F_n(w_n^{t,*}) + \nabla F_n(w^t)^\top w_n^{t,*}] \\
&\quad - \frac{1}{N}\sum_{n\in\mathcal{N}}\nabla F(w^t)^\top w^t + \frac{\beta}{2}\left\|\frac{1}{N}\sum_{n\in\mathcal{N}}(w_n^{t,*} - w^t)\right\|^2,
\end{aligned}
\tag{D.1}
$$

Then,

$$
\begin{aligned}
F(w^{t+1}) &\overset{(5.12)}{\leq} F(w^t) + \frac{1}{N\eta}\sum_{n\in\mathcal{N}}\{G_n^t(w_n^{t,*}) - F_n(w^t) \\
&+ [\nabla F_n(w^t) - \eta\nabla F(w^t)]^\top w^t\} + \frac{\beta}{2}\left\|\frac{1}{N}\sum_{n\in\mathcal{N}}(w_n^{t,*} - w^t)\right\|^2 \\
&\leq F(w^t) + \frac{1}{N\eta}\sum_{n\in\mathcal{N}}\{G_n^t(w_n^{t,*}) - F_n(w^t) \\
&+ [\nabla F_n(w^t) - \eta\nabla F(w^t)]^\top w^t\} + \frac{\beta}{2N}\sum_{n\in\mathcal{N}}\left\|w_n^{t,*} - w^t\right\|^2 \\
&\overset{(5.3)}{=} F(w^t) + \frac{1}{N\eta}\sum_{n\in\mathcal{N}}\{G_n^t(w_n^{t,*}) - G_n^t(w^t) \\
&- \frac{\alpha-\beta\eta}{2}\left\|w_n^{t,*} - w^t\right\|^2\} \\
&\overset{(5.3)}{=} F(w^t) - \frac{1}{N\eta}\sum_{n\in\mathcal{N}}\{\frac{\alpha-\beta\eta}{2}\left\|w_n^{t,*} - w^t\right\|^2 \\
&+ F_n(w^t) - F_n(w_n^{t,*}) + \nabla F_n(w_n^{t,*})^\top(w_n^{t,*} - w^t)\} \\
&\overset{(5.12)}{\leq} F(w^t) - \frac{1}{N\eta}\sum_{n\in\mathcal{N}}\frac{2\alpha-\beta\eta}{2}\left\|w_n^{t,*} - w^t\right\|^2 \\
&\overset{(5.11)}{\leq} F(w^t) - \frac{2\alpha-\beta\eta}{2N\eta\beta^2}\sum_{n\in\mathcal{N}}\left\|\nabla F_n(w_n^{t,*}) - \nabla F_n(w^t)\right\|^2 \\
&\overset{(5.7)}{=} F(w^t) - \frac{(2\alpha-\beta\eta)\eta}{2\beta^2}\left\|\nabla F(w^t)\right\|^2 \\
&\overset{(5.15)}{\leq} F(w^t) - \frac{(2\alpha-\beta\eta)\alpha\eta}{2\beta^2}[F(w^t) - F(w^*)].
\end{aligned}
\tag{D.2}
$$

Based on the above analysis, we have

$$
\begin{aligned}
F(w^t) - F(w*) &\leq [1 - \frac{(2\alpha-\beta\eta)\alpha\eta}{2\beta^2}][F(w^{t-1}) - F(w^*)] \\
&\leq ... \leq [1 - \frac{(2\alpha-\beta\eta)\alpha\eta}{2\beta^2}]^t[F(w^0) - F(w^*)] \\
&\leq e^{-t\frac{(2\alpha-\beta\eta)\alpha\eta}{2\beta^2}}[F(w^0) - F(w^*)].
\end{aligned}
\tag{D.3}
$$

We assign the global accuracy $\epsilon_g = e^{-t\frac{(2\alpha-\beta\eta)\alpha\eta}{2\beta^2}}$, i.e., $T = \frac{2\beta^2}{(2\alpha-\beta\eta)\alpha\eta}\ln(\frac{1}{\epsilon_g})$, the global FL problem is converged. Therefore, Lemma 9 is proved.

# BIBLIOGRAPHY


[1] J. Yao and N. Ansari, "QoS-aware fog resource provisioning and mobile device power control in IoT networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 167–175, Mar. 2019.

[2] W. Shi *et al.*, "Multi-drone 3-D trajectory planning and scheduling in drone-assisted radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8145–8158, Aug. 2019.

[3] J. Yao and N. Ansari, "QoS-aware power control in internet of drones for data collection service," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6649–6656, 2019.

[4] Y. Gu *et al.*, "Joint radio and computational resource allocation in IoT fog computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7475–7484, Aug. 2018.

[5] L. Silva, N. Magaia, B. Sousa, A. Kobusińska, A. Casimiro, C. X. Mavromoustakis, G. Mastorakis, and V. H. C. de Albuquerque, "Computing paradigms in emerging vehicular environments: A review," *IEEE/CAA J. Autom. Sin.*, vol. 8, no. 3, pp. 491–511, 2021.

[6] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 2019.

[7] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.

[8] J. Yao and N. Ansari, "QoS-aware rechargeable UAV trajectory optimization for sensing service," in *Proc. IEEE ICC 2019*, Shanghai, May 20-24, 2019.

[9] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1123–1152, 2016.

[10] M. Lu *et al.*, "Wireless charging techniques for UAVs: A review, reconceptualization, and extension," *IEEE Access*, vol. 6, pp. 29 865–29 884, 2018.

[11] X. Lu *et al.*, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 757–789, 2015.

[12] I. Krikidis, S. Timotheou, S. Nikolaou, G. Zheng, D. W. K. Ng, and R. Schober, "Simultaneous wireless information and power transfer in modern communication systems," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 104–110, Nov. 2014.

[13] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[14] I. Grondman *et al.*, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Cybern.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[15] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 88, pp. 16–27, 2018.

[16] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, and K. Huang, "A survey on smart agriculture: Development modes, technologies, and security and privacy challenges," *IEEE/CAA J. Autom. Sin.*, vol. 8, no. 2, pp. 273–302, 2021.

[17] Z. Ullah, F. Al-Turjman, and L. Mostarda, "Cognition in UAV-Aided 5G and beyond communications: A survey," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 3, pp. 872–891, Sep. 2020.

[18] N. Ansari, Q. Fan, X. Sun, and L. Zhang, "SoarNet," *IEEE Wirel. Commun.*, vol. 26, no. 6, pp. 37–43, 2019.

[19] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2430–2444, Oct. 2020.

[20] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS 2017*, Apr. 2017, pp. 1273–1282.

[21] J. Yao and N. Ansari, "Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control," *IEEE Internet Things J.*, 2020, doi: 10.1109/JIOT.2020.3022590, early access.

[22] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, H. V. Poor, and S. Cui, "Delay minimization for federated learning over wireless communication networks," *arXiv preprint arXiv:2007.03462*, 2020.

[23] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[24] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.

[25] P. K. Gopala, L. Lai, and H. El Gamal, "On the secrecy capacity of fading channels," *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4687–4698, 2008.

[26] Wenyuan Xu, Ke Ma, W. Trappe, and Yanyong Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network*, vol. 20, no. 3, pp. 41–47, 2006.

[27] P. Yang, X. Cao *et al.*, "Three-dimensional drone-cell deployment for congestion mitigation in cellular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9867–9881, Oct. 2018.

[28] F. Tang *et al.*, "AC-POCA: Anticoordination game based partially overlapping channels assignment in combined UAV and D2D-based networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1672–1683, Feb. 2018.

[29] Z. Gong *et al.*, "Design, analysis, and field testing of an innovative drone-assisted zero-configuration localization framework for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10 322–10 335, Nov. 2017.

[30] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[31] A. Koubaa and B. Qureshi, "Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet," *IEEE Access*, vol. 6, pp. 13 810–13 824, 2018.

[32] Y. Chen and L. Wang, "Privacy protection for internet of drones: A network coding approach," *IEEE Internet Things J.*, DOI: 10.1109/JIOT.2018.2875065, early access.

[33] S. Li *et al.*, "Joint admission control and resource allocation in edge computing for internet of things," *IEEE Netw.*, vol. 32, no. 1, pp. 72–79, Jan. 2018.

[34] A. Bader and M. Alouini, "Localized power control for multihop large-scale internet of things," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 503–510, Aug. 2016.

[35] J. Yao and N. Ansari, "Caching in energy harvesting aided internet of things: A game-theoretic approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3194–3201, Apr. 2019.

[36] G. Bacci *et al.*, "Energy-aware competitive power allocation for heterogeneous networks under QoS constraints," *IEEE Trans. Wireless Commun.*, vol. 14, no. 9, pp. 4728–4742, Sep. 2015.

[37] M. Mozaffari *et al.*, "Drone small cells in the clouds: Design, deployment and performance analysis," in *Proc. IEEE GLOBECOM 2015*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[38] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *Proc. IEEE GLOBECOM 2014*, Austin, Texas, Dec. 2014, pp. 2898–2904.

[39] J. Gundlach, *Designing Unmanned Aircraft Systems: A Comprehensive Approach*. American Institute of Aeronautics and Astronautics, 2012.

[40] D. Hulens, J. Verbeke, and T. Goedemé, "Choosing the best embedded processing platform for on-board UAV image processing," in *Proc. VISIGRAPP 2015*, Berlin, Germany, Mar. 2015, pp. 455–472.

[41] G. Auer *et al.*, "How much energy is needed to run a wireless network?" *IEEE Wireless Commun.*, vol. 18, no. 5, pp. 40–49, Oct. 2011.

[42] H. P. Benson, "Global optimization algorithm for the nonlinear sum of ratios problem," *J. Optimiz. Theory App.*, vol. 112, no. 1, pp. 1–29, Jan 2002.

[43] A. Zappone and E. Jorswieck, "Energy efficiency in wireless networks via fractional programming theory," *Found. Trends Commun. Inf. Theory*, vol. 11, no. 3-4, pp. 185–396, 2015.

[44] Y. Jong, "Practical global optimization algorithm for the sum-of-ratios problem," *arXiv preprint arXiv:1207.1153 [math.OC]*, 2012.

[45] J. B. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 1, pp. 181–217, 1960.

[46] Z. Zhou *et al.*, "When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5526–5538, Nov. 2018.

[47] K. Dorling *et al.*, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.

[48] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[49] R. Deng *et al.*, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[50] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[51] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[52] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[53] H. Y. Hwang *et al.*, "Modeling and analysis of an energy-efficient mobility management scheme in IP-based wireless networks," *Sensors*, vol. 11, no. 12, pp. 11 273–11 294, 2011.

[54] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[55] M. Wazid *et al.*, "Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3572–3584, Apr. 2019.

[56] B. Bera *et al.*, "Blockchain-envisioned secure data delivery and collection scheme for 5G-based IoT-enabled internet of drones environment," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9097–9111, Aug. 2020.

[57] J. Yao and N. Ansari, "Online task allocation and flying control in fog-aided internet of drones," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5562–5569, May 2020.

[58] D. Altinel and G. K. Kurt, "Modeling of hybrid energy harvesting communication systems," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 523–534, Jun. 2019.

[59] T. D. Nguyen, J. Y. Khan, and D. T. Ngo, "A distributed energy-harvesting-aware routing algorithm for heterogeneous IoT networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 4, pp. 1115–1127, Dec. 2018.

[60] A. M. Jawad *et al.*, "Wireless power transfer with magnetic resonator coupling and sleep/active strategy for a drone charging station in smart agriculture," *IEEE Access*, vol. 7, pp. 139 839–139 851, 2019.

[61] K. Lee and J. Hong, "Power control for energy efficient D2D communication in heterogeneous networks with eavesdropper," *IEEE Commun. Lett.*, vol. 21, no. 11, pp. 2536–2539, Nov. 2017.

[62] P. Mach and Z. Becvar, "Cloud-aware power control for real-time application offloading in mobile edge computing," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 5, pp. 648–661, May 2016. [Online]. Available: https://doi.org/10.1002/ett.3009

[63] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.

[64] P. Pace *et al.*, "Intelligence at the edge of complex networks: The case of cognitive transmission power control," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 97–103, 2019.

[65] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 2019.

[66] L. Lei *et al.*, "Multiuser resource control with deep reinforcement learning in IoT edge computing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 119–10 133, Dec. 2019.

[67] J. Yao and N. Ansari, "Caching in dynamic IoT networks by deep reinforcement learning," *IEEE Internet Things J.*, 2020, doi: 10.1109/JIOT.2020.3004394, early access.

[68] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3516–3526, Jun. 2019.

[69] J. Yao and N. Ansari, "Power control in internet of drones by deep reinforcement learning," in *Proc. IEEE ICC 2020*, Dublin, Jun. 7-11, 2020, pp. 1–6.

[70] M. Ku *et al.*, "Advances in energy harvesting communications: Past, present, and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1384–1412, 2016.

[71] R. G. Bernacki and N. Salamon, "Experimental study of energy harvesting in UHF band," in *J. Phys. Conf. Ser.*, vol. 709, 2016, p. 012009.

[72] J. Ho and M. Jo, "Offloading wireless energy harvesting for IoT devices on unlicensed bands," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3663–3675, 2019.

[73] B. Kiumarsi and F. L. Lewis, "Actor–critic-based optimal tracking for partially unknown nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 140–151, Jan. 2015.

[74] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, Third quater 2011.

[75] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[76] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[77] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX OSDI 2016*, Savannah, GA, USA, 2016, pp. 265–283.

[78] N. Hossein Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, Dec. 2016.

[79] X. Zhou, Q. Wu, S. Yan, F. Shu, and J. Li, "UAV-enabled secure communications: Joint trajectory and transmit power optimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4069–4073, Apr. 2019.

[80] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. ACM SAC 2017*, 2017, pp. 506–509. [Online]. Available: https://doi.org/10.1145/3019612.3019878

[81] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM 2019*, Apr. 2019, pp. 1387–1395.

[82] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[83] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, 2020, doi: 10.1109/TWC.2020.3037554, early access.

[84] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, Jul. 2020.

[85] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, Apr. 2020.

[86] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 2020.

[87] A. Mukherjee and A. L. Swindlehurst, "Detecting passive eavesdroppers in the MIMO wiretap channel," in *Proc. IEEE ICASSP 2012*, Mar. 2012, pp. 2809–2812.

[88] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[89] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3–4, pp. 231–357, Nov. 2015.

[90] Y. Zhou, C. Pan, P. L. Yeoh, K. Wang, M. Elkashlan, B. Vucetic, and Y. Li, "Secure communications for UAV-enabled mobile edge computing systems," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 376–388, 2020.

[91] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX HotCloud 2010*, Berkeley, CA, USA, 2010, pp. 4–4.

[92] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[93] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. Signal Process Sys.*, vol. 13, no. 2, pp. 203–221, Aug 1996.

[94] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[95] H. Ghazzai, M. Ben Ghorbel, A. Kadri, M. J. Hossain, and H. Menouar, "Energy-efficient management of unmanned aerial vehicles for underlay cognitive radio systems," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 4, pp. 434–443, 2017.

[96] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[97] V. Magnani, "Lipschitz continuity, aleksandrov theorem and characterizations for H-convex functions," *Mathematische Annalen*, vol. 334, no. 1, pp. 199–233, Jan. 2006.

[98] D. P. Bertsekas, *Nonlinear Programming*. Athena scientific Belmont, 1999.