Service Provisioning in Edge Computing for IoT Applications via Intelligent Resource Allocation and Optimization

Yuchen Li

October 2023

A thesis submitted for the degree of
Doctor of Philosophy of
the Australian National University

Australian
National
University

To everone I've met.

Declaration

This thesis is a presentation of the original work except where otherwise stated. I completed this work jointly with my supervisor, Professor Weifa Liang. My contribution to the work is around 80%. This research is partially funded by the Australian Research Council under its Discovery Project Scheme with Project No. DP200101985.

Yuchen Li
15 October 2023

Acknowledgments

I would like to acknowledge the following individuals and organizations for their invaluable support and contributions to the completion of my Ph.D. thesis:

First and foremost, I want to express my gratitude to my supervisor, Professor Weifa Liang, for his mentorship and support throughout my Ph.D. study. His expertise, insightful feedback and encouragement have been guiding me to be a competent researcher.

I want to express my appreciation to other members of my supervisor panel, Professor Hanna Kurniawati, and Professor Xiaohua Jia for their valuable inputs, constructive criticism, and suggestions. I also want to thank Dr Yu Lin who assisted with my PhD study and I am sorry that he cannot see my graduation. Meanwhile, I want to express my appreciation to Jing Li, Mengyu Chen and Yu Ma, for their collaboration, discussions and assistance.

I also want to express my appreciation to ANU School of Computing and its members for providing a stimulating academic environment and resources that facilitated my research. I also would like to acknowledge the financial support provided by ANU School of Computing. Their support through ANU Ph.D. scholarship has allowed me to focus on my research and pursue my academic goals. This thesis is also partially funded by the Australian Research Council under its Discovery Project Scheme with Project No. DP200101985.

I am grateful to my parents, Guihua Li and Ling Chen, for their encouragement and understanding throughout this challenging journey. Their support has been a constant source of motivation to me.

Last but not least, I want to thank my girlfriend, Zeruo Liu, for her support and accompanying me. I would like to spend the rest of my life with her, so I want to take the opportunity to ask a special question: Zeruo, will you marry me?

Publications

Journal

1. **Y. Li**, W. Liang, W. Xu, Z. Xu, X. Jia, Y. Xu and H. Kan. Data collection maximization in IoT sensor networks via an energy-constrained UAV. *IEEE Transactions on Mobile Computing*, Vol.22. No.1 pp. 159 – 174, 2023.

2. **Y. Li**, W. Liang, and J. Li. Profit driven service provisioning in edge computing via deep reinforcement learning. *IEEE Transactions on Network and Service Management*, Vol.19, No. 3, pp. 3006 – 3019, 2022.

3. **Y. Li**, W. Liang, J. Li, X. Cheng. D. Yu, A. Zomaya, and S. Guo. Energy-aware, device-to-device assisted federated learning in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, Vol.34. No.7, pp.2138–2154, 2023.

4. J. Li, W. Liang, **Y. Li**, Z. Xu, X. Jia, and S. Guo. Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism. *IEEE Transactions on Mobile Computing*, Vol.22, No.5, pp. 3017 –3030, 2023.

5. J. Li, W. Liang, W. Xu, Z. Xu, **Y. Li**, and X. Jia. Service home identification of multiple-source IoT applications in edge computing. *IEEE Transactions on Services Computing*, Vol. 16, No. 2, pp. 1417-1430, 2023.

Conference

1. **Y. Li**, W. Liang, J. Li, X. Cheng, D. Yu, A. Zomaya, and S. Guo. Energy-constrained D2D assisted federated learning in edge computing. *Proc of The 25th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile System* , ACM, pp.33–37, 2022.

2. **Y. Li**, W. Liang, and J. Li. Profit maximization for service placement and request assignment in edge computing via deep reinforcement learning. *Proc. of MSWiM'21*, ACM, 2021.

3. **Y. Li**, W. Liang, W. Xu, and X. Jia. Data collection of IoT devices using an energy-constrained UAV. *Proc of 34th IPDPS'20*, pp644–653, IEEE, 2020.

4. J. Li, W. Liang, **Y. Li**, Z. Xu, and X. Jia. Delay-aware DNN inference throughput maximization in edge computing via jointly exploring partitioning and parallelism. *Proc of IEEE 46th Conference on Local Computer Networks (LCN)*, IEEE, 2021.

5. M. Chen, W. Liang, and **Y. Li**. Data collection maximization for UAV-enabled wireless sensor networks. *Proc of 29th ICCCN'20*, IEEE, 2020.

6. J. Li, J. Wang, Q. Chen, **Y. Li**, and A. Zomaya. Digital twin-enabled service satisfaction enhancement in edge computing *Proc of IEEE INFOCOM'23*, IEEE, 2023.

Abstract

The Internet of Things (IoT), as an emerging technology that connects a wide range of smart devices to the Internet, has significantly impacted our daily lives and will continue to grow and evolve in the coming years. To meet delay-sensitive user service requirements, mobile edge computing (MEC) provides computing resources by placing cloudlets in the proximity of users. Also, with the development of MEC, an era of rapid growth and expansion in the fusion of Artificial Intelligence (AI) and IoT has begun. The increasing computation power allows the shift of computation and services from the remote cloud to the edge of the core network, which provides low-latency services for IoT devices. As a result, more and more sensors and smart devices are being deployed. Moreover, AI techniques help to manage limited computing, storage and communication resources in a dynamic MEC environment. In order to utilize the computation power of smart IoT devices while protecting user privacy, federated learning enables the training of deep neural networks at smart IoT devices and the aggregation of trained models at edge servers. Digital twins (DTs) as replicas of real objects, enable the provisioning of IoT services between the real world and the virtual digital world.

In this thesis, we study service provisioning for IoT applications in mobile edge computing networks via intelligent resource allocation and optimization. This introduces several challenges: (1) How to maximize the data collection from IoT devices deployed in remote areas by using energy-constrained unmanned aerial vehicles(UAVs); (2) How to synchronize digital twins (DTs) in cloudlets with their IoT devices in a real-time manner to minimize the DT state staleness under the limited synchronization budget per update round; (3) How to maximize the profit of IoT service providers by utilizing deep learning techniques to manage service placement on edge servers and allocation of user requests; (4) How to maximize the performance of deep neural networks by selecting federated learning clients in a training round in a device-to-device communication enabled edge environment. In this thesis, to tackle the aforementioned challenges, we identify several fundamental problems, and devise efficient algorithms for the problems. The contributions of the thesis are given as follows.

Firstly, we study the sensing data collection of IoT devices in a sparse IoT-sensor network, using an energy-constrained UAV, where the sensory data is stored in IoT devices while the IoT devices may or may not be within the transmission range of each other. We proposed a novel framework to enable the UAV to collect data from multiple devices simultaneously. We also devised efficient approximation and near-optimal algorithms to find the trajectory that maximizes the data collection.

Secondly, we consider the DT state staleness minimization problem of objects, where the state updating of digital twins (DTs) through synchronizing digital twins

with their physical objects in an MEC network, by a real-world application example: a UAV-enabled data collection in a renewable sensor network, and the collected data are then uploaded to their corresponding DTs in cloudlets in the MEC network for fidelity-aware user queries. We devise efficient algorithms and a data volume prediction mechanism for the problem. We also study on how to develop evaluation plans for queries on DT data.

Thirdly, we investigate the online service placement and request assignment problem in MEC networks, where service requests arrive one by one without the knowledge of future arrivals, and each arrived request demands a specific service with a tolerable service delay requirement, with the aim to maximize the profit of the service provider, through admitting as many service requests as possible for a given monitoring period. We implement a prediction mechanism for future request arrivals to pre-install service instances. We also develop an efficient request assignment algorithm.

Fourthly, we study energy-aware DNN model training in an edge computing environment. We formulate a novel federated learning framework that enables device-to-device communication when model uploading to mitigate communication overhead. We also develop near-optimal algorithms to minimize the loss of global models trained in the formulated framework, where the energy resources of devices and the bandwidth of the server are limited.

Finally, we summarise the work and discuss several potential research topics for future research based on the work in this thesis.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we briefly introduce basic concepts, models and problems in this thesis study. We will introduce mobile edge computing(MEC), the Internet of Things(IoT), edge intelligence and digital twins(DT). Then, we illustrate the system architecture proposed in the thesis. Also, we conduct a comprehensive investigation on service provisioning in edge computing for IoT applications via intelligent resource allocation, where we propose four research topics and pose the challenges on the proposed research topics. In the end, we summarize the contributions of this thesis and demonstrate the thesis organization.

1.1 Mobile Edge Computing

Recently, there has been a rapid growth of mobile devices and an explosion of mobile applications. However, some applications, such as face recognition, require a significant amount of computing resources, which are unlikely executed on mobile devices with limited processing power, memory, and battery life. To enable such computing-intensive tasks, mobile cloud computing (MCC) was proposed to solve this obstacle [14]. MCC integrates cloud computing services with mobile devices, where adequate computing resources and other services, such as data storage, are provided to mobile devices. The resource-intensive applications are then processed at remote data centers and provided to mobile devices as services. The energy consumption of mobile devices is reduced, which also prolongs their lifetime. As a result, mobile devices become more versatile and capable of performing a wider range of applications [106].

Despite the many benefits of MCC, there are some issues that need to be addressed due to that remote data centers are often located far away from mobile devices. Such long physical distances between the devices and the remote data centers lead to high processing latency [1]. Also, the communication data must be transferred via backhaul networks, which brings expensive communication costs and heavy load on the backhaul network [91]. To shorten the physical distance between the mobile devices and the resources, Mobile edge computing (MEC) brings computing resources closer to the edge of a network, i.e., closer to end users and devices [89]. More specifically, a number of edge servers with computing resources are

deployed at strategic locations in mobile edge networks, such as base stations and access points. Consequently, the processing tasks and services are moved from the remote data center to the proximity of user devices to reduce the end-to-end delay and the workload on backhaul networks while improving the performance of mobile applications [5].

MEC enables a wide range of applications and services that benefit from low-latency and high-bandwidth connections, such as augmented reality, virtual reality, video streaming, online gaming, and autonomous vehicles [10]. These applications have stringent requirements for end-to-end delay, while fast and responsive communication between mobile devices and computing resources can be achieved through MEC. On the other hand, MEC also improves network efficiency, enhances security and privacy, and increases availability and reliability of services [111]. By processing tasks at the edge of the network, the amount of data that needs to be transmitted via the backhaul network is reduced, thereby alleviating network congestion and reducing the load. The security and privacy of data are also enhanced by keeping sensitive data on-premise and reducing the exposure of data to the whole network. Moreover, the distributed architecture of MEC increases the availability and reliability of services. A failure in a centralized data center can lead to a complete service outage, whereas in MEC, the failures of edge servers will only affect limited areas.

The benefits of MEC can be summarised as follows:

- **Low latency:** By processing data closer to the edge of the network, MEC significantly reduces latency and improves response times. This is particularly important for applications that require real-time data processing, such as augmented reality, virtual reality, and autonomous vehicles.

- **Scalability:** MEC helps to improve the scalability of devices by distributing computing resources across multiple nodes at the edge of the network. This can help to reduce the load on centralized data centers and improve the overall performance of the system.

- **Security:** MEC provides enhanced security by processing data locally and reducing the need to transmit sensitive data over the network. This can help to protect against potential security breaches and ensure data privacy.

- **Location awareness:** MEC leverages location-based technologies to provide real-time information about the location and movement of devices and users, enhancing the user experience of location-based applications.

In summary, MEC can significantly improve the performance, efficiency, and security of many applications, making it a valuable technology for a wide range of applications.

1.2 The Internet of Things

The Internet of Things (IoT) is a rapidly growing technological paradigm that achieves connections among everyday devices and objects through the Internet, enabling communications between human-to-machine and machine-to-machine [90]. The communications allow smart devices to exchange data and interact with each other or with people, which generates vast amounts of data that can be analyzed to extend the frontier of knowledge, improving efficiencies for automation to revolutionize people's lives.

With wide deployments of smart IoT devices, these devices will generate and transfer a massive amount of data. It is predicted that by 2025 there will be over 80 billion devices connected to the Internet and the volume of generated data will reach 180 trillion gigabytes [107]. The data can be further analyzed for research studies or help organizations make decisions. For example, IoT devices can be used to monitor traffic patterns, optimize routes, and improve road safety. Meanwhile, IoT has the ability to automate processes based on the collected data by enabling the interactions between devices and devices [110]. For instance, in agriculture, IoT devices can monitor soil moisture levels, weather conditions, and crop health in real-time, and then make automatic adjustments to irrigation schedules and water usage based on the collected data.

Traditionally, IoT services are hosted in remote data centers. Therefore, massive data must be transferred from IoT devices to remote data centers for processing, which leads to a heavy burden to back-haul network and long latency. Also, many IoT applications have stringent latency requirements, which cannot be satisfied by MCC. As a promising technology, MEC, which provides computing and storage resources in the proximity of the IoT devices, enhances IoT by reducing the burden on the central cloud infrastructure and reducing the data transfer costs associated with sending large amounts of data to and from the cloud [115]. The end-to-end delay is also shortened due to the nearby computing resources, which unlocks potential applications of IoT services. Overall, the performance, capacity of IoT devices and the communication efficiency between devices is enhanced by MEC.

In summary, IoT has the following characteristics:

- **Connectivity:** IoT devices are connected to each other through a network, allowing them to exchange data and communicate with each other.
- **Sensing:** IoT devices can detect and measure changes in their environment, such as temperature, humidity, and motion.
- **Data processing:** IoT devices can process data locally or send it to a central location for processing, enabling real-time analysis and decision-making.
- **Automation:** IoT devices can be programmed to perform automated tasks, such as turning on lights when someone enters a room or adjusting temperatures based on occupancy.

1.3 Edge Intelligence

Deep Learning is a technique that utilizes deep neural networks to learn the patterns of a group of given data and make predictions or decisions accordingly [40]. With the advance of deep learning, smart services and intelligent applications, such as computer vision and natural language processing, have greatly impacted people's lives. There are also many deep learning based IoT applications, such as smart cities and autonomous vehicles. On the one hand, constrained by the limited resources of IoT devices, it is natural to offload computing-intensive deep learning tasks to the edge to shorten latency and prolong the lifetime of IoT devices [124]. On the other hand, the success of deep learning is not only attributed to the development of deep neural networks, but also the growing amount of data and computing resources [126]. The explosive increase in the number of mobile and IoT devices results in the phenomenal growth of the data volume generated at the edge of networks [7]. As a ceaseless data source, IoT devices provide huge amounts of data for deep neural network training. Therefore, edge intelligence, a paradigm where Artificial Intelligence and MEC are combined together, attracts much attention [32].

In spite of the adequate data source, the privacy of data is also a major concern [72]. Traditional training of deep neural networks needs users to upload their data to a centralized server, which has a high risk of privacy violation. Federated learning, as an emerging framework, provides a solution to utilizing the data on training while protecting privacy [3]. In Federated Learning, Smart devices train DNN models locally, using their local data. The trained local models are periodically sent to a server for aggregation, and the aggregated global model is then sent back to the devices for further training. In the FL paradigm, users only upload their locally trained models rather than their collected data to the server, which protects the privacy of users.

In addition to the benefits to deep learning, edge intelligence also enables intelligent resource optimization and allocation for service provisioning in edge environments. Conventional edge resource optimization and allocation depend on the assumption that the network is static. However, due to the mobility and heterogeneity of IoT, deep learning instead, is promised to learn the patterns of IoT services and manage the edge networks with its reasoning ability, resulting in an efficient resource allocation under dynamic edge networks [36].

In summary, edge intelligence mainly studies on the following aspects [126]:

- **DL application at edge** This studies on how to schedule limited resources of edge to host applications of deep neural networks to meet different requirements such as end-to-end delay.

- **DL training at edge:** This studies on how to conduct deep neural network training at edge environments, by facilitating data collection from devices and applying distributed machine learning techniques under computing and communication constraints.

- **DL for optimizing Edge** This studies on how to utilize deep learning techniques to smartly allocate the resources in mobile edge networks, such as request assignment or service instance placement.

1.4 Digital Twins

In recent years, the term "digital twin" has been gaining popularity and interest, which is emerged as a powerful tool for design, simulation, monitoring, and optimization of physical systems [135]. A digital twin refers to a virtual representation of a physical object in a virtual environment, allowing for real-time monitoring and analysis of its behavior by synchronizing the DT (the virtual representation) with physical objects. The wide adoption of MEC, IoT and edge intelligence elevates the development of DTs. The universally deployed IoT sensors enable the collection of data of any physical objects, so that the DTs can simulate their real physical objects better. Also, with the computing and storage resources provided by MEC, DTs are able to synchronize with the physical objects and process queries with short latency. Moreover, the advance of edge intelligence also provides DTs with powerful data analysis abilities. The DTs then perform decisive predictions on the behaviors of the physical system in different scenarios, identify potential issues before they arise, and optimize the performance of monitored objects.

Due to the accuracy and real-time, DTs can be used in many areas, such as manufacturing, transportation, and healthcare. The DT market is forecast to reach $15.66 billion by 2023 at an annual growth rate of 37.87% according to a market research in 2017 [33], and more organizations and institutions have gained a lot of interest in DTs. For instance, GE developed a DT platform - PREDIX that can better understand and predict asset performance [105]. SIEMENS's focus covers smart operations during the complete process of product design, production and operation [34]. The wide adoption of DTs will further be fuelled up by strong service demands from various users in the edge of networks, such as Internet of Things (IoT), autonomous driving, healthcare, smart city, AR/VR, and so on. The study of DTs and their applications is still in the early stage, accelerating the usage of digital twin technologies will bring unprecedented benefits in service provisioning, intelligent resource allocation of service for 5G wireless networks, and 6G network management, through real-time digital representations of physical objects [37, 54, 95, 114].

1.5 System Architecture

In this thesis, we consider a remote sensor network and a mobile edge network. The remote sensor network consists of a UAV, a depot, and several IoT sensors, where the IoT sensors generate data through sensing and the UAV is used to collect data from the IoT sensors. The UAV recharges and offloads its collected data at a depot. The collected data will then be transmitted to the cloudlets in the mobile edge networks. The mobile edge network consists of a set of access points (AP), and some

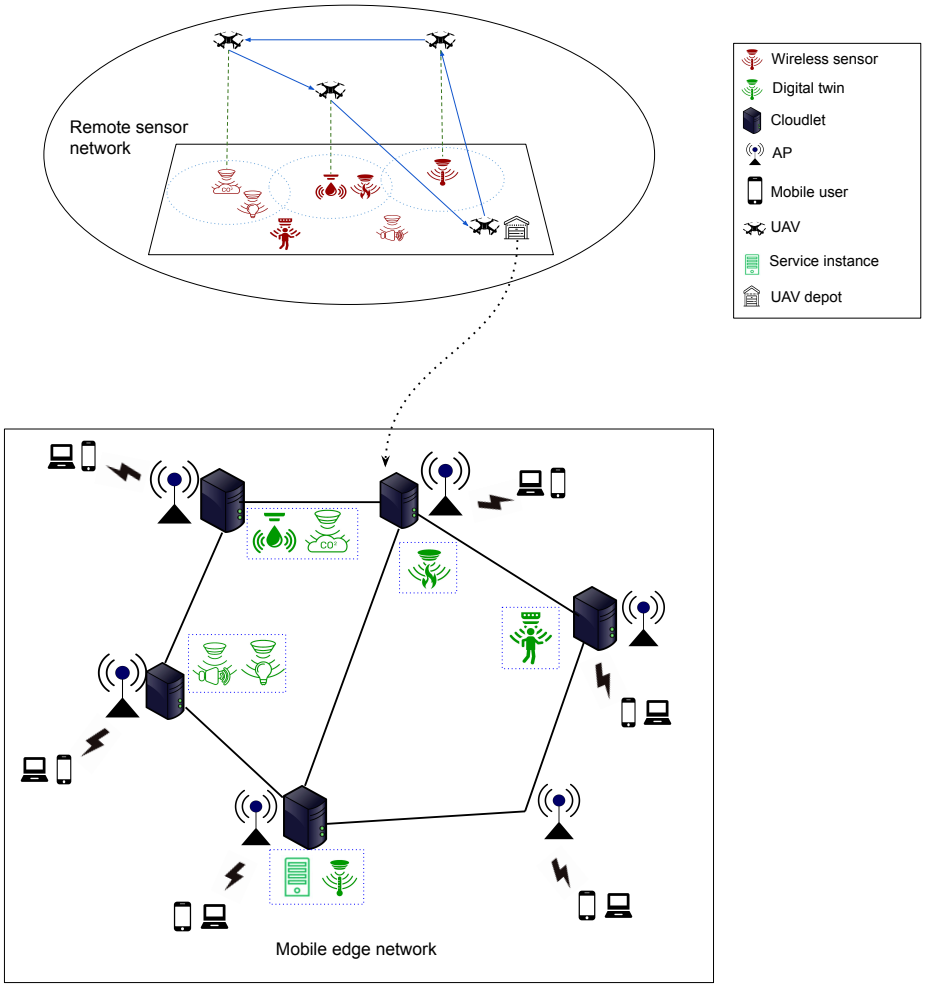


Figure 1.1: An illustrative example of an MEC network and a remote sensor network.

resource-constrained cloudlets are co-located with some of the APs. We assume that the APs and their co-located cloudlets are connected via an optical link with high capacity, and thus the bandwidth on the link is not considered as a constraint and the transmission delay over the link can be ignored. The cloudlets host IoT service instances and DTs corresponding to the IoT sensors in the remote sensor network. Also, users can request IoT and DTs services through the AP within its transmission range. Fig. 1.1 is an illustrative example of the system architecture discussed in the thesis.

1.6 Research Topics

To enable mobile edge networks to provide low-latency and diverse IoT services to a growing number of IoT devices and users, the thesis focuses on the following

four main research topics:

(1) How to maximize the total data collection of IoT devices in a sparse IoT-sensor network, using an energy-constrained Unmanned Aerial Vehicle (UAV), where the sensory data is stored in IoT devices while the IoT devices may or may not be within the transmission range of each other.

(2) How to minimize the staleness of digital twins (DTs) through synchronizing DTs with their physical objects in a mobile edge computing (MEC) network under a fixed update budget by predicting the volume of data generated by physical objects, and how to develop a cost-effective evaluation plan to conduct fidelity-aware queries on the DT data.

(3) How to utilize deep learning techniques to maximize the profit of service providers when service requests arrive one by one without the knowledge of future arrivals, and each arrived request demands a specific service with a tolerable service delay requirement through admitting as many service requests as possible for a given monitoring period.

(4) How to minimize the global loss of federated learning in mobile edge environments where the energy capacity and bandwidth are constrained.

These topics are interrelated with each other. Topic (1) discusses how to maximize the data collection when using an energy-constrained UAV to collect data from IoT devices. Topic (2) in the sequel studies the data synchronization of DTs, where the data synchronization between real objects in remote sensor networks and DTs placed in MEC networks is achieved by a UAV-enabled data collection. While in Topic (1) we assume that the volume of data to be collected is given, we remove this assumption by using deep learning to predict the generated data volume in Topic (2). While topic (3) investigates how to utilize deep reinforcement learning to dynamically place IoT service instances and offload IoT service requests to appropriate cloudlets, the DT technique studied in topic (2) provides abilities to monitor, analyze, and collect data to the prediction mechanism proposed in topic (3). As a follow-up to topic (3), topic (4) considers how to minimize the loss of federated learning by enabling model uploading via D2D communication, which increases the performance of the prediction mechanism in topic (3).

In the following, we briefly introduce our research topics and summarize the related work on the research topics.

1.6.1 Data Collection Maximization via an Energy-constrained UAV

The use of mobile charging vehicles or mobile data collection vehicles on the ground for sensor charging and sensory data collection has been widely studied in the past [74, 75, 86, 140, 139, 138, 141]. Due to its high flexibility, low cost and ease of deployment, Unmanned Aerial Vehicle (UAV) has been a key enabling technology that has received significant attention recently, and it has been widely applied in

civilian environments such as natural disaster rescuing, good deliveries, crop health assessment, and so on [43]. In this thesis, we study sensory data collection from IoT devices on the ground using a UAV. Specifically, we consider a sparse sensor network that consists of many IoT devices for sensing their surroundings. Some of the IoT devices serve as *aggregate sensor nodes* to store their own and neighbors' sensing data. The stored data at an aggregate sensor node will then be collected periodically by a UAV for further processing. As the volume of data stored at different aggregate sensor nodes is significantly different, the hovering durations of the UAV for data collection at different hovering locations are different, and the amounts of energy consumed by the UAV at different hovering locations thus are different. In addition, it does incur energy consumption when the UAV travels from one hovering location to another hovering location. As the energy capacity of the UAV is given, this poses a great challenge. That is, how to find a closed tour for the UAV including its depot for data collection such that the total volume of data collected by the UAV at different hovering locations in the tour is maximized, subject to its energy capacity. Furthermore, how long the UAV should stay at each of the hovering locations in the tour to ensure that all (or partial) data stored at the IoT sensor devices covered by it will be collected?

There are several existing studies focused on finding the trajectories of charging or data collection for one or multiple mobile vehicles. However, due to various obstacles in the sensor networks including buildings, ponds, rivers block roads in the monitoring region, mobile vehicles cannot travel in the region for sensor charging or data collection smoothly. In recent years, there has been a growing interest in the employment of UAVs for sensory coverage and data collection for wireless sensor networks, as UAVs have freedoms for data collection while avoiding the mentioned obstacles by flying over them [97, 76, 98]. For example, Mozaffari *et al.* [97] considered trajectory path findings for multiple UAVs with the aim to minimize the total transmission energy consumption of IoT devices to upload their data to the UAVs, where the UAVs are treated as aerial base stations. They proposed a clustering method to cluster IoT devices into different clusters and then find trajectories for multiple UAVs that sojourn only at the cluster centers. However, the clustering method only minimizes the sum of distances between each sojourn location of the UAV and device locations, it does not consider the distances among sojourn locations. Zhan *et al.* [153] jointly considered working states of sensors and the trajectory of the UAV, by utilizing a fading channel model for the senor-UAV links to minimize the maximum energy consumption of sensors. Ghorbel *et al.* [39] focused on the identification of hovering locations of a single UAV for data collection from a cluster of sensors with the aim to minimize the energy consumption of the UAV. Liang *et al.* [76] considered a coverage quality problem via a UAV. They assumed that the hovering time of the UAV at each hovering location is identical, for which they proposed an approximation algorithm for the coverage quality maximization problem. Binol *et al.* [11] aimed at finding time-optimal paths for multiple UAVs for data collection. Zhan and Zeng [152] studied the time minimization problem of data collection, using multiple UAVs. Sikeridis *et al.* [112] proposed a novel frame-

work that improves the energy efficiency in UAV-supported Public Safety Networks. Guo *et al.* [41] considered the problem of a fleet of UAVs for disaster surveillance by finding trajectories for the UAVs such that the longest tour time among the UAVs is minimized. Recently Zhang *et al.* [155] investigated how to minimize the number of UAV deployments when the tour of each UAV is bounded by a given budget, for which they proposed constant approximation algorithms for the problem under different data collection models. Samir *et al.* [109] considered data collection with data uploading deadlines, where a UAV is dispatched for data collection from time-constrained devices and each device has a data uploading deadline, they aimed to maximize the network throughput. Chen *et al.* [17, 18] considered the data collection maximization problem without considering the traveling energy consumption of a UAV, they devised approximation algorithms for the problem, they also considered the data collection under different data transmission rate by proposing a heuristic algorithm [19].

The essential differences between the work in the thesis and the aforementioned studies lie in the following two aspects. One is that we consider the partial or full data collection from multiple IoT devices simultaneously by the deployment of a UAV that has not been considered previously. Another is that an approximate solution for the problem without hovering coverage overlapping is developed, which is the first algorithm with a provable performance guarantee for the problem.

1.6.2 Digital Twin Synchronizations and Fidelity-aware Query Services in Edge Computing

In the past decade, DTs have gained significant impetus as a breakthrough technological development that has the potential to transform the landscape of manufacturing today and tomorrow. As updating DT states of physical objects incurs costs, we assume that the update (synchronization) budget at each update round is fixed, this implies that not all DT states of physical objects are stored in cloudlets but only some of them can be updated at each update round. Thus, it is crucial to determine the DT states of which physical objects to be updated at each update round in order to minimize the average staleness of DT states of all physical objects.

In this thesis, we consider a fundamental DT state synchronization problem through an application example of a renewable sensor network for monitoring a remote area, where each sensor is powered by a solar panel and its data generation is determined by its energy. We assume that for each sensor, there is a corresponding DT deployed in a cloudlet in the MEC network, and the data collected by the sensor will be uploaded at a time slot, which indicates the DT state of the sensor will be updated at that time slot. We further assume that the update budget at each update round is to dispatch an energy-limited UAV for data collection from sensors, with the aim to minimize the average DT state staleness of all objects for a finite time horizon, where the finite time horizon can be divided into equal time slots. We will answer the following questions: how long should a UAV hover above a sensor for its data collection? what is the amount of data generated by a sensor since its

last data collection? at which time slots should the sensory data of a sensor be collected or synchronized with its DT? how to update the DT states of sensors such that the average staleness of DT states of all sensors is minimized under the limited synchronization budget? and what is the relationship between the staleness of DT states and the fidelity of a query result built upon the DT data? We will address these mentioned challenges.

MEC emerged as a promising paradigm for delay-sensitive IoT applications. Empowered by DTs technology, MEC platforms can provide services for various delay-aware IoT and AI applications [37, 54, 77, 83, 95]. For example, Dong *et al.* [37] focused on a deep learning-based model training by mapping an MEC network to its DTs network, and proposed an efficient algorithm for the DT network training with the aim to minimize the training energy consumption. Lin *et al.* [77] devised an incentive-based congestion control scheme to meet dynamic service demands of DTs in an MEC network, by utilizing the Lyapunov optimization technology. Lu *et al.* [83] developed a federated learning algorithm based on the blockchain technique to enhance data privacy and security in DTs-assisted MEC networks.

There are also several existing studies that focus on optimizing either service delays or data freshness in DT-enabled MEC networks [29, 84, 114, 125]. For example, Corneo *et al.* [29] studied the problem of timely dissemination of sensor updates to clouds that provide DTs service for users with the aim to minimize the age of information (AoI). They devised a novel push-and-pull method of sensor information updating to strive for a non-tradeoff between sensory data freshness and the tolerable query delay. Sun *et al.* [114] utilized DTs to minimize the offloading latency, through adopting the Lyapunov optimization technique. Lu *et al.* [84] devised a deep learning-based algorithm to cope with the mobility of mobile devices and migration of DTs, with the aim to minimize the average service delay.

The study of this thesis is closely related to the AoI that usually is to minimize the average duration of information from their generation to their usage in diverse applications [9, 81, 127, 146, 156]. Since the AoI is very sensitive to the transmission and processing delays of information since its generation, most existing studies of AoI are conducted in MEC platforms, due to the MEC networks being closer to the sources (sensory devices) of information generation. For example, Liu *et al.* [81] studied the AoT problem by developing an approximation algorithm for finding multiple routing paths between a pair of source and destination nodes in a network and showed that the problem is equivalent to minimizing the maximum-delay one among the routing paths between the source and destination nodes. Bastopcu and Ulukus [9] considered an information update system, where a receiver requests updates from an information provider to minimize its age of information, and the updates are generated at the information provider by completing a set of collecting data and performing computation tasks, subject to a minimum required quality (maximum allowed distortion) constraint on the updates. They developed an efficient policy for the problem to achieve their optimization objective. Wang *et al.* [127] devised an efficient offline scheduling algorithm and an online learning algorithm for minimizing the average Age of Critical Information (AoCI) of mobile clients. Xu *et*

al. [146] dealt with the AoI minimization for IoT big data applications, by devising an online learning method for dynamic request admissions. Zhang *et al.* [156] proposed an efficient algorithm to minimize the average service delay while meeting content freshness requirements, by striving for a non-trivial trade-off between the AoI and the service delay for timely content refreshing.

Although there are some similarities between the average AoI of data generated by physical objects and the average staleness of DT states of physical objects, the DT technology is more powerful and complicated, compared with the AoI technique. The DT technology not only concentrates on minimizing the average AoI of DT states of physical objects but also emphasizes how to make use of the DT states to activate or predict the behaviors of physical objects in the future, while the AoI technology does not.

Unlike the aforementioned studies of DT-enabled applications in MEC networks, to the best of our knowledge, we are the first to introduce a fundamental problem in DT-enabled MEC networks – the budget-constrained DT state synchronization problem, by proposing a novel framework for it. We also show how to utilize the staleness (or freshness) of DT states of physical objects to perform fidelity-aware queries on physical objects through an example of UAV-enabled data collection for a remote, renewable sensor network.

1.6.3 Profit Driven Service Provisioning via Deep Reinforcement Learning

MEC has been envisioned as a promising solution to provide adequate computing resources with high Quality of Service (QoS) through deploying cloudlets (edge servers) within the proximity of users [65]. Users can offload their delay-sensitive tasks to cloudlets instead of remote clouds for services, thereby reducing not only end-to-end service delays but also the communication resource consumption of core networks [137]. Combining MEC with virtualization techniques such as Network Function Virtualization (NFV), service providers now can rent out cloudlet resources for users by implementing user demanded services as service instances for profits [58]. For example, Augmented Reality (AR) devices are usually resource-constrained, AR service providers can place service instances at cloudlets to assist the processing of AR programs with low latency. On the other hand, AI services can be provided at cloudlets, where smart turnstiles can upload photos of visitors to MEC in order to verify visitor identities. Through these examples, it can be seen that different applications request different services. Meanwhile, the locations and arriving rates of different service requests are correlated, which usually follow spatial-temporal correlations. For instance, AR services are frequently requested from the locations such as museums or memorial halls while smart turnstiles usually request face identification services during rush hours.

In this thesis, we will develop such a prediction mechanism to predict future service requests and their requested services. We will also devise an efficient algorithm for service instance placements built upon the prediction results, and we finally will perform service request assignments to cloudlets to maximize the profit

of the service provider. To this end, it poses several challenges.

- First, we are only allowed to install a limited number of service instances at each cloudlet. Since different services incur different operational costs and different amounts of computational resource consumption, it is challenging to decide how many service instances of each type of services should be installed at cloudlets in order to maximize the profit of the service provider. On the other hand, although the removal of an idle NFV instance can reduce the operational cost of the service provider, determining the removal of which idle NFV instances is challenging, as this can be illustrated by an extreme example, where newly incoming requests demand just removed service instances of a service, these newly arrived requests may not be admitted due to long instantiation delays on their demanded service instances. Also, the instantiation cost must be considered if it frequently happens to install and remove service instances from the system.
- Second, predicting future service types and service demands is difficult. Inaccurate prediction incurs a large initialization cost and unnecessary computing resource consumption. Therefore, without the knowledge of future service request arrivals and service types, judiciously placing different types of service instances at resource-limited cloudlets is challenging.
- Finally, it is challenging to assign service requests to different cloudlets in order to maximize the number of requests admitted, by taking into account service delay requirements of these requests, capacities of cloudlets, and the accumulative profit of admitted requests.

Several studies on service provisioning and Virtual Network Function (VNF) instance placement in MEC environments have been conducted in the past. For example, Xu *et al.* [145] jointly considered the placement of VNF instances for data processing and data traffic routing path planning for user requests in a multi-tier edge cloud network to maximize network throughput. They devised an efficient algorithm for request admissions, and an approximation algorithm for the problem without end-to-end delay requirements. Meanwhile, they dealt with the mobility of mobile devices by adopting machine-learning methods. Sun *et al.* [113] considered placing VNFs of Service Function Chains (SFC) to cloudlets and connecting VNFs in the SFC in order, with the aim to minimize the total placement cost. They proposed a time-efficient algorithm based on affiliation-aware VNF placement for an offline-version of the problem. They also utilized the Fourier-Series-based prediction method to predict the demands of VNFs for the online-version of the problem. Cziva *et al.* [31] studied the dynamic VNF placement problem to minimize the end-to-end latency of requests, considering network dynamics, user resource demands and user mobility. They developed a scheduler that delivers an optimal solution based on the optimal stopping theory [13]. Kayal and Liebeherr [53] considered service placement in a fog network to minimize the energy consumption and communication cost. They devised a fully distributed service placement algorithm based

on the game theory. Li *et al.* [62] optimized the placement of SFCs to maximize the total expected profit under an assumption that both demanded computing resource and traffic data rates are uncertain. They proposed a near-optimal approximation algorithm by adopting the Markov approximation technique. He *et al.* [44] focused on the service placement and request scheduling while considering sharable and non-shareable resources in an MEC network with the aim to maximize the number of requests admitted. Yousefpour *et al.* [151] introduced a novel framework for QoS-aware dynamic fog service provisioning, where services can be dynamically deployed or released in fog nodes. They proposed two efficient greedy algorithms to minimize the cost while meeting the QoS requirements of applications. Hassan *et al.* [42] studied the service placement problem with the aim to reduce the response time for critical services and the energy consumption for non-critical services. They proposed an efficient service placement algorithm that considers the priority of services, network latency, and the power efficiency jointly. Ning *et al.* [99] studied the dynamic service placement framework where mobile users move erratically, with the aim to maximize the system utility. They first decomposed the long-term optimization problem by utilizing the Lyapunov optimization method. They then adopted a sample average approximation-based stochastic algorithm to approximate the future expected system utility. They finally determine service placement configurations by giving a distributed Markov-based approximation algorithm.

There are also investigations on the online service placement and request assignment problem of service provisioning in MEC networks. For example, Du *et al.* [38] investigated the problem of request offloading and content caching. They formulated the problem as a stochastic optimization problem with the aim of maximizing the profit, by jointly optimizing the computation offloading strategy, resource allocation, and Internet content caching strategy. They formulated a Lyapunov optimization algorithm for the problem. Li *et al* [59] considered a VNF provisioning problem to maximize the revenue, by admitting user requests with service reliability requirements. They proposed two online algorithms with provable competitive ratios for the problem, by adopting the primal-dual technique. Yang *et al.* [149] aimed at maximizing the profit of cloud providers by admitting a subset of requests and finding optimal scheduling of user requests. Ma *et al.* [87] studied the profit maximization problem in an MEC network by dynamically admitting delay-aware requests while meeting SFC requirements. Samanta *et al.* [108] focused on the optimization of the profit and latency. They proposed an adaptive service offloading scheme that maximizes the revenue collected while maintaining network utility. Xu *et al.* [142] considered the assignment of requests to shared VNF instances to maximize network throughput while minimizing the operational cost. They proposed a prediction mechanism to dynamically adjust the number of VNF instances needed in cloudlets, which later serves as a baseline in our experimental section. Liu *et al.* [78] studied the optimization of both new and existing user SFC deployments with the aim to maximize the profit, while considering the resource consumption and operational overhead. They proposed an integer linear programming (ILP) solution, and presented a column generation (CG) solution to reduce the time complexity of the

algorithm.

There are several recent studies that utilize deep learning for task offloading and resource allocation problems in MEC networks [45, 79]. For instance, Tang *et al.* [119] considered task offloading in an MEC environment, where they utilized the long short-term memory (LSTM), dueling deep Q-network (DQN), and double-DQN techniques to decide whether to offload tasks, and to which edge node each offloading task should be offloaded. Huang *et al.* [49] considered task offloading and wireless resource allocation in an MEC under dynamic wireless network conditions. They proposed a Deep Reinforcement learning-based Online Offloading (DROO) framework, which achieves near-optimal performance with low computation time. Liu *et al.* [80] make use of deep reinforcement learning method to predict the locations of servers, and obtain an optimal task offloading decision in a Vehicle Edge Computing, where moving vehicles act as mobile edge servers to provide computation services. Min *et al.* [94] studied task offloading of IoT devices with energy harvesting. They proposed a reinforcement learning based scheme to decide the destination edge device of task offloading and the offloading rate. Chen *et al.* [24] proposed a double deep Q-network (DQN)-based strategic computation offloading algorithm to find the optimal task offloading destination base station, where the deep neural networks can predict the network dynamics in prior. However, the above studies mainly use deep learning as a solution to combinatorial optimization problems, or they utilize deep learning to predict statuses of a network, such as the network conditions or the workload of servers.

Unlike the aforementioned studies that determine service placements when service requests are given in advance, in this thesis we focus on placing service instances prior to service request arrivals, which is much more difficult as there is no way to know when and how many requests of each type of services arrive in advance. We will adopt the deep reinforcement learning method as a prediction mechanism to achieve this.

1.6.4 Energy-Aware, Device-to-Device Assisted Federated Learning

Deep learning is a technique that utilizes deep neural networks to learn patterns of a group of data. The rapid development of deep learning has led to desperate demands on a large volume of data for more accurate predictions. Traditional training of deep neural networks needs users to upload their data to a centralized server, which has a high risk of privacy violation and lots of communication bandwidth consumption. Federated learning (FL), as a promising framework, provides a solution to utilizing the data on training while protecting privacy. In Federated Learning, Smart devices train DNN models locally, using their local data. The trained local models are periodically sent to a server for aggregation, and the aggregated global model is then sent back to the devices for further training. In FL paradigm, users only upload their locally trained models rather than their collected data to the server, which protects the privacy of users.

As devices usually are powered by limited batteries, performing federated learn-

ing consumes lots of energy. A typical way is to train the model on an edge server, and each device uploads its local model to the server for aggregation, and finally a global model is formed in the edge server. Therefore, MEC, which brings computing resources to the edge of the core network so that the data generated by mobile devices can be processed at the edge to reduce the burden of the back-haul network, has emerged as a promising paradigm for federated learning in the era of the Internet of Things [103, 144, 148]. However, devices with longer distances from the edge server need to use larger transmission powers to upload their local models, thereby consuming much more energy than those of devices with shorter distances. Also, training a large-scale DNN model needs a large volume of data transmissions between devices and the edge server, this introduces a heavy communication overhead [50]. Due to the limited energy imposed on devices, not every device has sufficient energy to train its local model on its dataset. Instead, only a subset of the dataset is used for local model training due to the energy budget of the device for training. Consequently, the solution accuracy and convergence of federated learning training might degrade because not all data are used for the model training.

There are extensive studies on federated learning in edge computing environments. Some of them focused on limited network and computation resource allocations [35, 116, 134, 131], while others concentrated on client choices [20, 100, 147]. There were several investigations on the learning parameters of federated learning, including the frequency of local updates and global aggregations [131, 154]. Unlike the above studies, in this thesis, we introduce a device-to-device (D2D) assisted uploading concept to alleviate the communication overhead on uploading local models from devices to the edge server. That is, devices in edge environments are paired according to their distances and energy availability. For each pair of devices, the one with less energy budget sends its trained local model to another with a more energy budget, and the received device then aggregates the model with its local model and uploads the aggregated model to the edge server. Since the wireless bandwidth capacity at the edge server is limited, such a model aggregation and uploading can reduce the volume of data transmitted to the edge server. Performing energy-aware federated learning in edge environments thus poses several challenges. First, by allocating more energy on computation, a device can train its local model on a large volume portion of its collected dataset, but this leaves the device less energy on its local model uploading or vice versa, how to strive for a non-trivial trade off of energy allocations between its local model training and local model transmission/uploading? Second, due to the heterogeneity of computing power and volume of collected data, different devices have different amounts of energy budgets at different rounds, the selection of device pairings heavily impacts not only the transmission energy consumption of devices but also the amount of data for local model training, thereby affecting the accuracy and convergence of federated learning, how to pair devices such that the energy consumption is minimized while the convergence can be guaranteed is challenging. Finally, the wireless bandwidth capacity of the edge server usually is bounded, which can support only up to K devices rather than all devices to upload their local models to the server simultaneously. As the contributions to-

wards the global model of different devices are different, some devices are more important than others. Then, how to identify top contribution devices to upload their local models to the edge server? We will tackle the aforementioned challenges in this thesis.

Federated Learning in edge computing environments has been extensively investigated recently. Most studies investigated energy consumption on devices and edge servers, others dealt with the non-trivial trades off between the communication cost, the accuracy of solutions, and the convergence speeds of different learning algorithms. For example, Wang *et. al* [131] focused on the trade-off between the communication cost and the convergence performance, they provided an upper bound of FL convergence that later is used to develop an approximation algorithm for the FL problem, with the aim to minimize the loss function under resource capacity constraints. Dinh *et. al* [35] concentrated on the trade-off between energy consumption and model convergence. They proposed a new FL algorithm (FEDL) with the assumption of strongly convex and smooth loss functions. The crux of their algorithm is a new local surrogate function for each device to train its local model approximately up to a local accuracy level. Sun *et. al* [116] considered a long-term energy-aware dynamic edge server scheduling problem. They developed an online scheduling algorithm based on the Lyapunov optimization, with the aim to maximize the average number of edge server participants in training at each training round. Tu *et. al* [130] studied federated learning in a fog computing environment, where devices are allowed to offload their local data to other devices for processing and discard some of their data. They investigated the impact of data transfer between devices and data discarding on model training. Chen *et. al* [20] dealt with the optimization of user selections and network resource allocation in a wireless network for a set of users and a base station, with the aim to minimize the convergence time of federated learning. They developed a scheme to select users with high contributions to the convergence of the training, and formulated an integer linear programming for network resource allocation. Zhang *et. al* [154] considered a scenario where the data across different clients are non-independent and identically distributed (Non-IID), for which they proposed a deep reinforcement learning based method to decide the batch size of local training adaptively. Nishio and Yonetani [100] studied the federated learning within the deadline of the global aggregation. They proposed a federated learning protocol to select as many participants as possible from a set of heterogeneous devices with limited resources. Wu *et. al* [134] dealt with reducing the traffic volume of WAN transmissions. They proposed a hierarchical federated learning paradigm, which performs synchronous federated learning aggregation between clients and edge servers and asynchronous aggregation between edge servers and cloud servers. They also devised algorithms to decide the federated learning controllable factors, staleness threshold client-edge association, and data distribution. Chen *et. al* [22] studied serverless decentralized federated learning, where workers only communicate with their neighbors. They enabled the exchange of intermediate results instead of the entire model between the workers. They also proposed an efficient algorithm to trade off between the communication cost and training performance. Lu *et. al* [82]

tackled a heterogeneity problem of local data, by exploiting a clustered FL framework and an auction-based client selection strategy with constrained local resources. Pilla [103] aimed to minimize the energy consumption of FL training on heterogeneous devices. The author proposed an optimal pseudo-polynomial solution to find the optimal workload distribution. Chen *et. al* [16] aimed to mitigate the communication overhead of transferring DNN models, and proposed a novel framework that identifies and freezes stable parameters of DNN models to improve communication efficiency. Tang *et. al* [120] sparsified the DNN model in decentralized federated learning to deal with the communication bottleneck, and devised an algorithm to find a peer selection that efficiently utilizes bandwidth resources. Tao *et. al* [121] proposed a Byzantine-resilient distributed gradient descent algorithm for FL that can handle the heavy-tailed data and converge under the standard assumptions. They also developed another algorithm to further reduce the communication overhead on learning process, using the gradient compression technique, and theoretical analysis shows that the proposed algorithms can achieve the statistical error rate that is order-optimal. Xu *et. al* [143] investigated the problem of energy minimization for a single FL request with uncertain availability of UEs, by proposing a novel optimization framework. They also devised an online learning algorithm with a bounded regret for the UE selection, by considering various contexts (side information) that influence energy consumption. Xu *et. al* [144] proposed a hierarchical FL framework for joint worker aggregator placement and UE assignment for a single FL request, and devised an approximation algorithm for it. They also considered the online worker aggregator placement and UE assignment for multiple FL request admissions with model uncertainty, by proposing a multi-armed bandit based online learning algorithm for it.

The aforementioned studies mainly investigated resource allocation or client (user) selection for a set of devices that upload their local models to a server for aggregation, by striving for non-trivial trade-offs between energy consumption and accuracy of the solution obtained. However, none of these works considered using energy of neighbor devices of a device for its local model uploading. To the best of our knowledge, we are the first to study how to minimize the global loss of a model through D2D transmissions. The saved energy at each device can be used for local model training to further improve the performance of federated learning.

1.7 Thesis Contributions

The main contributions of this thesis are conducting a comprehensive study on intelligent service provisioning and resource allocation in mobile edge environments. For the aforementioned research topics, we propose novel frameworks, and develop system models for the proposed framework. We then formulate optimization problems and devise near-optimal and efficient algorithms.

The contributions of this thesis are summarized as follows.

- For the problem of data collection of IoT devices in a sparse IoT-sensor net-

work, we study on how to maximize the data collection by using a UAV in Chapter 2. Specifically, we consider an energy-constrained UAV to collect data from all IoT devices within its transmission range, and the energy consumption on UAV hovering and flying does not exceed its battery capacity. We devise efficient approximation and heuristic algorithms to find the tour of the UAV when it fully or partially collects data from IoT devices. The contributions of this research topic are: (1) a novel framework where a UAV collects data from multiple IoT devices simultaneously, (2) approximation algorithms for the full and partial data collection problems when transmission coverage at different hovering locations of UAV has no overlapping and (3) efficient heuristic algorithms for the full and partial data collection problems when transmission coverage at different hovering locations of UAV has overlapping.

- For the problem of budget-constrained DT state synchronizations, we aim to minimize the staleness of DT in Chapter 5. We study a real application scenario, where an energy-constrained UAV is dispatched to collect data from sensors to update the status of the DT objects. We then investigate how to develop a cost-effective evaluation plan for a fidelity-aware query to DT objects deployed in a mobile edge network. The contributions involved in this topic include: (1) a novel staleness minimization problem of DT states of sensors for a finite time horizon, under a fixed synchronization budget per update round, (2) an optimal algorithm for a special case of the staleness minimization problem, (3) an efficient algorithm for the staleness minimization problem, (4) a deep learning mechanism to predict the volume of generated data by each sensor and (5) a cost-effective evaluation algorithm for fidelity-aware queries to DT objects.

- For the problem of service placement and request assignment in an MEC network, we study on the maximization of the profit of service providers in Chapter 3. We place and remove IoT service instances at resource constrained cloudlets in advance of the arrival of requests to shorten the end-to-end delay of requests in dynamic mobile edge networks without the knowledge of future arrivals. The contributions of this research topic are: (1) a framework that pre-installs and revokes idle IoT service instances in a dynamic service placement, (2) a deep reinforcement learning prediction mechanism to predict which service requests arrive in future, and pre-install the demanded service instances accordingly, (3) and a request assignment that offloads requests to appropriate cloudlets.

- For the problem of D2D-assisted federated learning in mobile edge environments, we investigate how to minimize the loss of global models in Chapter 4. To mitigate communication overhead, we formulate a novel framework that allows devices to send their models to their neighborhoods for local aggregation. We also consider energy capacity of devices and bandwidth constraints on an edge server, where we only allow a limited number of devices to communicate

with the server simultaneously. The contributions of this research topic are: (1) A novel energy-constrained D2D-assisted federated learning framework where devices with limited energy utilize neighbor devices for model uploading, (2) an near-optimal algorithm for a special case of the problem, and (3) a heuristic algorithm for the problem.

It is also worth noting that the formulated frameworks, devised algorithms and analysis techniques, will be of independent interest in many other domains, such as artificial intelligence and combinatorial optimization.

1.8 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 studies the data collection in sensor networks via an energy-constrained UAV. Chapter 3 tackles the DT state staleness minimization problem, subject to the limited synchronization budget. Chapter 4 investigates the profit maximization of service providers by service instance placement and request assignment. Chapter 5 consider federated learning in a D2D-enabled, energy constrained edge environment, with the aim to minimize the loss of a global model. Chapter 6 summarizes the thesis and discusses future works.

Chapter 2

Data Collection Maximization via an Energy-Constrained UAV

In this chapter, we formulate two novel data collection problems to fully or partially collect data stored from IoT devices using the UAV subject to the energy capacity on the UAV. To this end, we propose a novel data collection framework that enables the UAV to collect sensory data from multiple IoT devices simultaneously. We then formulate two data collection maximization problems to deal with full or partial data collection from IoT devices at each hovering location, and show that both defined problems are NP-hard. We instead devise approximation and heuristic algorithms for the problems.

2.1 Introduction

With the increasing popularity of Internet of Thing devices, more and more applications of smart homes/smart cities, e-health care, and artificial transportations built upon IoT devices become part of our daily life [67]. However, most IoT devices usually have very limited energy, computational and storage capacities due to their portable sizes [66]. Sometimes, it is unrealistic to allow these devices to transmit or relay sensing data to a base station through multihop relays, due to the significant transmission energy consumption, and in the worst case, they may not be in the transmission ranges of each other. It is very challenging to collect sensing data from these IoT devices for processing on time. Due to its accessibility, versatility and flexibility, UAVs are considered as a promising solution to IoT data collection.

The novelties of this chapter lie in the provisioning of a novel framework of data collection from multiple IoT sensor devices simultaneously, via an energy-constrained UAV. We formulate two data collection maximization problems and develop efficient approximation and heuristic algorithms for the problems that strive for a fine tradeoff between the energy usages of the UAV on hovering and traveling, respectively. To the best of our knowledge, this is the first time that the use of a UAV for full or partial data collection from multiple IoT devices simultaneously is studied. Efficient algorithms for finding a data collection trajectory for the UAV are developed.

The rest of the chapter is organized as follows. Section 2.2 introduces the system model, notions, and notations. Section 2.3 presents the problem definitions and shows the NP-hardness of the defined problems. Section 2.4 devises approximation algorithms for the full and partial data collection maximization problems without hovering coverage overlapping when the problem size is large. Section 2.5 proposes efficient heuristic algorithms for the full and partial data collection maximization problems with hovering coverage overlapping. Section 2.6 evaluates the proposed algorithms empirically, and Section 2.7 concludes the chapter.

2.2 Preliminaries

In this section, we first introduce the system model, notions and notations.

2.2.1 System model

We consider an IoT application scenario where many different types of IoT devices are deployed in a given region for monitoring purposes, some of the IoT devices (sensors) are chosen as *aggregate sensor nodes* that can store both their own sensing data and their neighbors' sensory data, assuming those IoT devices that have not been chosen as aggregate sensor nodes can forward their sensory data to one of their neighboring aggregate sensor nodes. In case there are multiple aggregate sensor neighbors, it can choose one of such neighbors for the storage of its sensory data. Since aggregate sensor nodes are sparsely distributed, they may or may not be within the communication range with each other. The sensory data collected at each aggregate sensor node thus cannot be transferred to the base station through multi-hop relays, or there are obstacles between the aggregate sensor nodes, e.g., ponds, or buildings that prevent the relays. Furthermore, there are energy-constrained on aggregate sensor nodes as relaying data will consume their considerable amounts of energy. To prolong the lifetime of each aggregate sensor node, a UAV is deployed for data collection from the aggregate sensor nodes. We assume that the UAV with a constant speed v is at a depot d initially and powered by a limited energy battery $\mathcal{E}$. The UAV consumes energy at hovering locations for data collection from aggregate sensor nodes in its hovering coverage range and traveling (flying) from one hovering location to another hovering location. For the sake of convenience, in the rest of this chapter we term the aggregate sensor nodes as IoT devices or sensor nodes interchangeably if no confusion arises. The aggregate sensor nodes in a monitoring region form a sparse sensor network $G = (V \cup \{d\}, E)$, where V is the set of aggregate sensor nodes, and there is an edge $e \in E$ between each pair of aggregate sensor nodes. The depot of the UAV is d in which the UAV will be recharged and its collected data will be downloaded for further processing.

To ensure that the UAV can return depot d per tour, its data collection tour must be a closed tour including depot d. The duration of a tour of the UAV will be determined by the tour length and the volume of data stored in the IoT devices covered by the UAV at each hovering location in the tour. Assume that the UAV

takes T time units to finish its tour, in which T_h and T_t are the amounts of time spent on hovering and traveling respectively, then $T = T_h + T_t$ and the total amount of energy consumed by the UAV in the tour must meet that $T_h \cdot \eta_h + T_t \cdot \eta_t \leq \mathcal{E}$, where η_h and η_t are the energy consumption rates of the UAV on hovering and traveling, respectively [117].

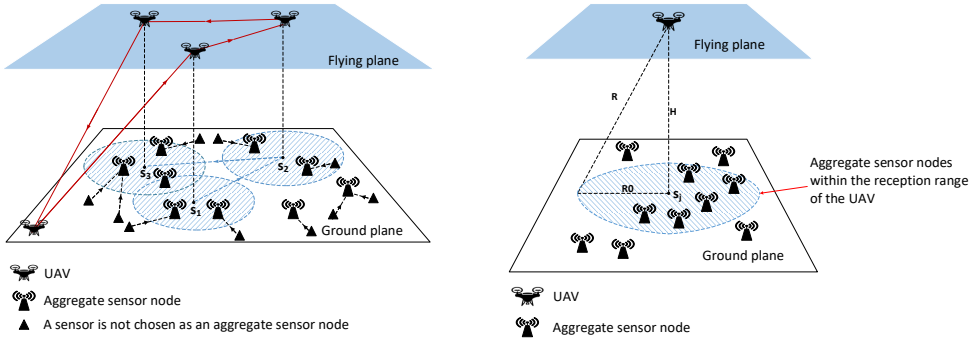
2.2.2 Data collection framework from IoT devices using a UAV

We assume that each aggregate sensor node in a monitoring region is labeled by its coordinates $(x_i, y_i, 0)$. Denote by (x', y', H) the coordinates of a hovering location of the UAV, where H is the flying altitude of the UAV, which is no greater than the transmission range R of each aggregate sensor node, and B is the data transmission rate of any aggregate sensor node. An aggregate sensor node $v \in V$ can transmit its stored data to the UAV with the data transmission rate B if the UAV is within its transmission range R. The hovering altitude H of the UAV for data collection thus is no greater than R, i.e., $H \leq R$, and we further assume that the altitude H of the UAV does not change [4].

Following the data collection model, if all IoT devices are within the reception range of the UAV, their transmitted data can be collected by the UAV, assuming that each IoT device uses a different communication channel [97]. We assume that the reception range of the UAV is a ball centralized at its hovering location, this ball is projected to the ground where the IoT devices are located to form a circle with the same radius of the ball, the data of all aggregate sensor nodes within the projected circle can be collected by the UAV when it hovers at the center of the ball, through the use of the orthogonal frequency division multiple access (OFDMA) technique [97].

Since there are infinite hovering locations for the UAV in the sky, to make the problem tractable, we assume that the hovering locations of the UAV are finite, by partitioning its hovering region into finite numbers of equal squares with edge length $\delta > 0$. In the rest of the discussion, we assume that the measurement unit of the UAV movement in its hovering region is the edge length δ of each square. Or the coordinates of potential hovering locations within a square are indistinguishable. For the sake of convenience, we further assume that the center of each square is the potential hovering location of the UAV within that square area. For a given data collection period T, assume that the volume D_v of data stored at each aggregate sensor node $v \in V$ is given, which consists of its own sensory data and the data forwarded by its neighboring IoT devices that have not been chosen as aggregate sensor nodes. Fig. 2.1 is an illustrative example of an aggregate sensor network G with a UAV for data collection.

We assume that the hovering region of a UAV is partitioned into M squares: $s_1, s_2, \ldots, s_M$, and the UAV performs data collection at the centers of these squares $s_j = (x_j, y_j, H)$ as its hovering locations, let $C(s_j)$ be the set of aggregate sensor nodes whose distances to the projected location $(x_j, y_j, 0)$ of s_j on the ground are no more than the data reception range R_0 of the UAV, i.e., the data stored at an aggregate senor node $v_i \in V$ with coordinates $(x_i, y_i, 0)$ can be collected by the UAV if it is within the



(a) An IoT network (b) A hovering location of the UAV

Figure 2.1: An example of an IoT sensor network G via a UAV for data collection.

ball of the UAV centered at hovering location s_j, i.e., $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_0$ and $R_0 = \sqrt{R^2 - H^2}$ (see Fig. 2.1 (b)). For any two potential hovering locations $s_i \in M$ and $s_j \in M$ with $i \neq j$, if $C(s_i) \cap C(s_j) = \emptyset$, we refer to the set M as a set of hovering locations *without hovering coverage overlapping*; otherwise, we refer to the set M as a set of hovering locations *with hovering coverage overlapping*. Notice that the data transmission duration of an aggregate sensor node from the ground to the UAV usually is determined by its distance to the UAV, its data volume, and its data transmission rate. Given two aggregate sensor nodes at different locations in the coverage circle of the UAV, it is well known that their data transmission durations and the data transmission rate will be different even if they have the same amounts of data to be transmitted. However, such differences between them are negligible if the UAV altitude H is relatively low (not too high). For the sake of discussion simplicity, in this chapter we assume that all sensors within the hovering coverage range of the UAV will have the same data transmission rate B.

The hovering (sojourn) duration of the UAV for data collection at hovering location s_j thus is

$$t(s_j) = \max_{v_i \in V} \{ \frac{D_{v_i}}{B} \mid \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_0 \}, \tag{2.1}$$

the total volume of data collected at hovering location s_j is

$$V(s_j) = \sum_{v_i \in V} \{ D_{v_i} \mid \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_0 \}, \tag{2.2}$$

and the total amount of energy consumption on data collection by the UAV at s_j is

$$w_1(s_j) = t(s_j) \cdot \eta_h. \tag{2.3}$$

2.2.3 Approximation ratio

Given an instance of a maximization optimization problem $\mathcal{P}$, let A and OPT be a feasible solution delivered by an approximation algorithm and the optimal solution of the problem respectively if $\frac{A}{OPT} \geq \frac{1}{\alpha}$ with $\alpha > 1$, then $\frac{1}{\alpha}$ is referred to the approximation ratio of the approximation algorithm for $\mathcal{P}$.

2.3 Problem formulations and NP-hardness of the defined problems

In this section, we define two novel data collection maximization problems using a single UAV, under the assumptions that (i) sensory data from multiple IoT devices can be collected simultaneously by a UAV at a hovering location; and (ii) the data stored at each IoT device can be either fully collected or partially collected by the UAV at each hovering location. Both problems are defined as follows.

Definition 1. Given an aggregate sensor network $G(V \cup \{d\}, E)$ and a UAV with energy capacity $\mathcal{E}$, each aggregate sensor node $v \in V$ has a volume D_v of data for collection, *the full data collection maximization problem* is to find a closed tour for the UAV such that the accumulative volume of data collected by the UAV at all hovering locations in the closed tour is maximized, subject to the energy capacity $\mathcal{E}$ on the UAV, assuming that the data stored at each IoT device in the hovering coverage of the UAV will be fully collected.

We deal with the full data collection maximization problem under two different restrictions. That is, whether the hovering coverage ranges of the UAV at any two different hovering locations are allowed to be overlapping. We thus have two cases for the problem: *the full data collection maximization problem without hovering coverage overlapping* and *the full data collection maximization problem with hovering coverage overlapping*, respectively.

Sometimes, the UAV may not need to collect all stored data of IoT devices at its hovering location, this may help the UAV to save energy for collecting more data from other IoT devices when it hovers at other hovering locations. We here use an example (see Fig. 2.2) to illustrate this scenario. Assume that the UAV can stop at two hovering locations s_1 and s_2 with hovering coverage overlapping. We further assume that it takes 10 minutes and 6 minutes to collect all data when it is located at s_1 and s_2, respectively. If it is allowed to collect partial data from the IoT devices when it is located at s_1 and s_2, for example, it takes 5 minutes to collect partial data at s_1 and 6 minutes to collect partial data at s_2. The same amount of data will be collected from both hovering locations in the end. However, its total energy consumption on full data collection at the two locations is $(10+6)\eta_h$ energy units, while the total energy consumption on the partial data collection at the two locations is $(5+6)\eta_h$ energy units, thereby saving energy of the UAV. Motivated by this example, given a positive integer $K \geq 1$, we can partition the sojourn duration $t(s_j)$ of the UAV at each hovering location s_j into K equal sojourn durations:

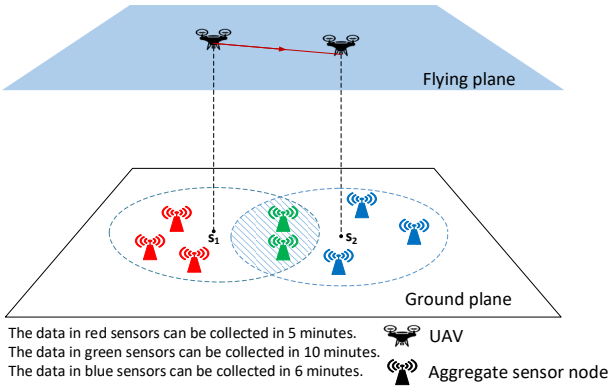


Figure 2.2: An illustration of partial data collection in an IoT sensor network G via a UAV.

$t(s_j)/K$, $2 \cdot t(s_j)/K$, $\ldots$, $K \cdot t(s_j)/K$, respectively. In other words, for each hovering location $s_j \in S$, there are K corresponding *virtual hovering locations* $s_{j,1}, s_{j,2}, \ldots, s_{j,K}$ with sojourn durations $t(s_j)/K$, $2 \cdot t(s_j)/K$, $\ldots$, $K \cdot t(s_j)/K$, respectively. The maximum amount $V(s_{j,k})$ of data collected by the UAV at a virtual hovering location $s_{j,k}$ with sojourn duration $k \cdot t(s_j)/K$ is

$$V(s_{j,k}) = \sum_{v \in C(s_j)} \{\frac{B \cdot k \cdot t(s_j)}{K} \mid \frac{D_v}{B} \geq k \cdot t(s_j)/K\} + \sum_{v' \in C(s_j)} \{D_{v'} \mid \frac{D_{v'}}{B} < k \cdot t(s_j)/K\}, \tag{2.4}$$

where $C(s_j)$ is the set of IoT devices within the hovering coverage range of the UAV at hovering location s_j, and the IoT devices in $C(s_j)$ can further be partitioned into two subsets: their data transmission time is no less than $k \cdot t(s_j)/K$, and strictly less than $k \cdot t(s_j)/K$ with $1 \leq k \leq K$, respectively.

$$t(s_{j,k}) = k \cdot t(s_j)/K. \tag{2.5}$$

That is, the amount of data collected by the UAV at each hovering location s_j can be from the partial volume of data to the full volume of data, i.e., $V(s_{j,1}) \leq V(s_{j,2}) \leq \ldots \leq V(s_{j,K})$ with $t(s_{j,1}) < t(s_{j,2}) < \ldots < t(s_{j,K})$. We define this partial data collection via a UAV as follows.

Definition 2. Given an aggregate sensor network $G(V \cup \{d\}, E)$ and a UAV with energy capacity $\mathcal{E}$ located at a depot d initially, each IoT device $v \in V$ has a volume D_v of data for collection, *the partial data collection maximization problem* in G is to find a closed tour including the depot d for the UAV such that the accumulative volume of data collected by the UAV at hovering locations in the tour is maximized, subject to the energy capacity $\mathcal{E}$ of the UAV, assuming that the UAV is allowed to collect partial data at each hovering location.

It can be seen that the full data collection maximization problem is a special case of the partial data collection maximization problem when $K = 1$, and unfortunately, both problems are NP-hard, which are stated by the following theorem.

Theorem 2.1. *Both the full data collection maximization problem and the partial data collection maximization problem in an aggregate sensor network $G(V \cup \{d\}, E)$ are NP-hard.*

Proof. We show that the full data collection maximization problem without hovering coverage overlapping is NP-hard, by a reduction from a well-known NP-hard problem - the orienteering problem [123]. We consider a special case of the data collection maximization problem where the potential hovering location of the UAV is on top of each aggregate sensor node, and there is not any energy consumption on data collection at each hovering location. We further assume that there is not any hovering coverage overlapping between any two hovering locations. Even for this special data collection maximization problem, we show that it is equivalent to an orienteering problem in G as follows.

Given a node- and edge-weighted, indirect graph $G(V, E)$, in which each node $v \in V$ has a positive award $p(v) = D_v$ and each edge $(u, v) \in E$ has a positive integral length $\frac{l(u,v)\eta_t}{\nu}$, and a given integral length L, the orienteering problem is to find a closed tour in G including a specified node (the depot d) such that the total award collected from the nodes in the closed tour is maximized, subject to the tour length no greater than L [8, 12, 123].

We reduce the orienteering problem in G to this special full data collection maximization problem as follows. The award collected at each hovering location $s_j \in S$ (the coordinates of s_j are (x_j, y_j, H) assuming that the coordinates of $v_j \in V$ are $(x_j, y_j, 0)$) is $V(s_j) = D_{v_j}$, $L = \lceil \frac{\mathcal{E}}{\eta_t} \rceil$, and the hovering energy consumption at each potential hovering location $s_j \in S$ ($S = V$ by the assumption)is zero. As the orienteering problem is NP-hard [123], the data collection maximization problem is NP-hard.

The full data collection maximization problem is a special case of the partial data collection maximization problem when $K = 1$, the latter is NP-hard, too. □

2.4 Approximation Algorithms for a special case

In this section, we deal with the full and partial data collection maximization problems without hovering coverage overlapping. We first formulate an Integer Programming solution for the problem when the problem size is small or moderate. We then propose approximation algorithms for the problems under the assumption that different hovering locations of the UAV within each square are indistinguishable. We finally analyze the correctness of the solutions and the time complexity of the proposed algorithms.

2.4.1 ILP formulation

In the following we formulate an integer programming solution (ILP) to the full data collection maximization problem without hovering coverage overlapping. Given

the aggregate sensor network $G = (V, E)$ and the UAV, we partitioned the hovering region of the UAV into M squares, denoted by $S = \{s_1, s_2, ..., s_M\} \cup d$. Let x_j be the indicator variables which determine whether the UAV will hover at the square s_j and collect the data, and let $y_{i,j}$ be the indicator variables which determine whether the UAV flies from square s_i to s_j, and $y_{0,i}$ and $y_{j,0}$ indicates if the UAV flies from the depot d to s_i and if the UAV flies from s_j to depot d respectively, where $1 \leq i, j \leq M$ and $i \neq j$. We formulate the ILP solution as follows:

$$\text{Maximize} \quad \sum_{j=1}^{M} V(s_j) \cdot x_j \tag{2.6}$$

subject to:

$$(2.1), (2.2)$$

$$x_j = \sum_{i=0}^{M} y_{i,j} \tag{2.7}$$

$$\sum_{i=1}^{M} y_{i,j} \leq 1, \quad \forall j \tag{2.8}$$

$$\sum_{j=1}^{M} y_{i,j} \leq 1, \quad \forall i \tag{2.9}$$

$$\sum_{i=0}^{M} y_{i,j} = \sum_{i'=0}^{M} y_{j,i'} \quad j \neq 0 \tag{2.10}$$

$$\sum_{i=0}^{M} \sum_{j=0}^{M} y_{i,j} \cdot \frac{l(s_i, s_j)}{\nu} \cdot \eta_t + \sum_{j'=1}^{M} t(s'_j) \cdot \eta_h \leq \mathcal{E} \tag{2.11}$$

$$\sum_{i=1}^{M} y_{i,0} = 1 \tag{2.12}$$

$$\sum_{j=1}^{M} y_{0,j} = 1 \tag{2.13}$$

$$\sum_{s_i, s_j \in S'} y_{i,j} \leq |S'| - 1, \forall S' \subseteq S \setminus \{d\}, S' \neq \emptyset \tag{2.14}$$

$$x_j \in \{0, 1\} \quad \forall s_j \in S \tag{2.15}$$

$$y_{i,j} \in \{0, 1\} \quad \forall s_i, s_j \in S \cup \{d\}, \tag{2.16}$$

where $V(s_j)$ and $t(s_j)$ are defined in Eq. (2.2). and Eq. (2.1), $l(s_i, s_j)$ is the Euclidean distance between s_i and s_j.

The objective (2.6) is to maximize the sum of profits of nodes on the data collection tour. Constraint (2.7) restricts that the UAV can only hover at s_j if s_j is on the flying path. Constraints (2.8) and (2.9) ensure that each hovering location can be reached or left by once. Constraint (2.10) ensures that if the UAV arrives at node s_j,

it must left from node s_j. Constraint (2.11) enforces that the sum of the energy consumption is no greater than the UAV energy capacity $\mathcal{E}$. Constraints (2.12) and (2.13) enforce that the UAV must start from the depot d and end at d. Constraint (2.14) prevents subtours disconnected from the depot d. Constraints (2.15) and (2.16) restrict the ranges of decision variable x_j and $y_{i,j}$ to be either 0 or 1.

2.4.2 Approximation algorithm for the full data collection maximization problem problem without hovering coverage overlapping

The basic idea behind the proposed algorithm is to reduce the problem to *the orienteering problem* [123]. The challenge of such a reduction lies in that the hovering energy consumptions of the UAV at different hovering locations are different, we aim to find a closed tour including depot d such that the accumulative volume of data collected by the UAV at all hovering locations in the tour is maximized, while the total amount of energy consumed of the UAV on both hovering and traveling is no greater than its energy capacity $\mathcal{E}$. We address this challenge by finding a closed tour in an auxiliary graph in which each edge is assigned an energy weight for both hovering at the endpoints of the edge and traveling along the edge as follows.

Since there are infinite numbers of potential hovering locations in the hovering plane of the UAV, to enable the problem to be tractable, we partition the hovering region of the UAV (or the corresponding IoT device deployment region) into a number of squares with edge length $\delta > 0$. We assume that the distance differences among potential hovering locations in each square are negligible if the value of δ is sufficiently small. We further assume that the center of each square is a potential hovering location for the UAV when it is in the square.

Having partitioned the hovering region of the UAV into M squares, we now construct a node and edge weighted, undirected graph $G_s = (S, E_s;\ p(\cdot), w_1(\cdot), w_2(\cdot,\cdot))$ as follows. S is the set of potential hovering locations of the UAV, and E_s is the set of edges that the UAV hovers and flies from one hovering location to another hovering location. The functions related to nodes and edges in G_s are defined as follows. $p : S \mapsto \mathbb{R}^{\geq 0}$ is the award function, $w_1 : S \mapsto \mathbb{R}^{>0}$ is the hovering energy consumption function, and $w_2 : E_s \mapsto \mathbb{R}^{\geq 0}$ is the energy consumption function on both hovering and traveling. There is an edge $(s_i, s_j) \in E_s$ between each pair of nodes s_i and s_j in S.

For each potential hovering location $s_j \in S$, the award (the amount of data collected) is

$$p(s_j) = \sum_{v_i \in C(s_j)} D_{v_i}, \tag{2.17}$$

where $C(s_j)$ is the set of IoT devices in the hovering coverage range of the UAV when it hovers at hovering location s_j, i.e., $C(s_j) = \{v_i \mid v_i \in V \ \&\ \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_0\}$, assuming that the UAV is at location $s_j = (x_j, y_j, H)$. If $C(s_j) = \emptyset$, then $p(s_j) = 0$, $t(s_j) = 0$ and $w_1(s_j) = 0$. D_{v_i} is the volume of data stored at IoT device v_i that is a function of the monitoring duration T and the sensing data generation rates of

neighboring sensors of v within the monitoring period.

The hovering duration $t(s_j)$ of the UAV at hovering location s_j for collecting data from IoT devices in $C(s_j)$ is

$$t(s_j) = \max_{v_i \in C(s_j)} \{\frac{D_{v_i}}{B}\}, \tag{2.18}$$

where B is the data transmission rate of an aggregate sensor node (an IoT device).

The amount of energy consumption of data collection by the UAV at hovering location s_j thus is

$$w_1(s_j) = t(s_j) \cdot \eta_h. \tag{2.19}$$

We assign a weight $w_2(s_j, s_k)$ to each edge $(s_j, s_k) \in E_s$ as follows.

$$w_2(s_j, s_k) = \frac{w_1(s_j) + w_1(s_k)}{2} + \frac{l(s_j, s_k) \cdot \eta_t}{\nu} \tag{2.20}$$

where the first term in the right hand side of Eq. (2.20) is half the sum of the amounts of hovering energy consumed by the UAV for data collection at locations s_j and s_k respectively, the second term is the amount of traveling energy consumption of the UAV along edge (s_j, s_k), and $l(s_j, s_k)$ is the Euclidean distance between locations s_j and s_k.

Having the constructed auxiliary graph $G_s(S, E_s;\ p(\cdot),\ w_1(\cdot),\ w_2(\cdot,\cdot))$, the orienteering problem in G_s is to find a closed tour including depot d such that the total award collected from the hovering locations in the tour is maximized, subject to that the tour length (measured in terms of energy) is no greater than the energy capacity $\mathcal{E}$ of the UAV. It can be seen that a solution to the orienteering problem in G_s returns a solution to the full data collection maximization problem in G without hovering coverage overlapping.

The detailed algorithm for the full data collection maximization problem is given in `Algorithm` 1.

2.4.3 Approximation algorithm for the partial data collection maximization problem without hovering coverage overlapping

We now deal with the partial data collection maximization problem by reducing the problem to the orienteering problem as well, and by adopting the proposed algorithm `Algorithm` 1 with minor modifications. Specifically, for a given integer $K \geq 1$ and each potential hovering location of the UAV, the K virtual hovering locations are created for the potential hovering location, i.e., K virtual hovering locations $s_{j,1}, s_{j,2}, \ldots, s_{j,K}$ are generated for each potential hovering location $s_j \in S$. Recall that $t(s_j)$ the hovering duration at s_j for all sensor data collection of sensors under this hovering coverage, we divide this duration into K equal time durations, and assign a

Algorithm 1 Approximation algorithm for the full data collection maximization problem without hovering coverage overlapping

Input: An aggregate sensor network $G = (V, E)$ with a set V of aggregate sensor nodes, a UAV with energy capacity $\mathcal{E}$ at depot d, and each node $v \in V$ has data volume D_v, and a given constant $\delta > 0$ but $\delta \leq R_0$.

Output: Find a closed tour including the depot d for the UAV such that the volume of data collected from all aggregate sensor nodes covered by the UAV at hovering locations in the tour is maximized, subject to the energy capacity of the UAV.

1: Partition the monitoring region into M squares $s_1, s_2, \ldots, s_M$ with the edge length of each square being δ; let $S = \{d, s_1, s_2, \ldots, s_M\}$;
2: Compute $t(s_j)$, $p(s_j)$, and $w_1(s_j)$ for each $s_j \in S$ with $1 \leq j \leq M$;
3: Construct an auxiliary graph $G_s = (S \cup \{d'\}, E_s \cup \{(v, d') \mid (v, d) \in E_s\}; p(\cdot), w_1(\cdot), w_2(\cdot, \cdot))$, where d' is a dummy depot;
4: Find a simple path P in G_s between the depot d and the dumpy depot d' such that the total award collected in the path is maximized (as there is no coverage overlapping between any two hovering locations by the assumption), subject to the energy capacity $\mathcal{E}$ of the UAV, by the approximation algorithm for the orienteering problem in metric graphs [8];
5: **return** A closed tour C derived from P for the UAV, which contains the hovering locations and the sojourn time at each of the hovering locations.

hovering duration $t(s_{j,k}) = \frac{t(s_j)}{K}$ at virtual hovering location $s_{j,k}$, i.e.,

$$t(s_{j,k}) = \frac{t(s_j)}{K}, \quad \forall k \in \{1, 2, \ldots, K\} \tag{2.21}$$

The volume $V'(s_{j,k})$ of data collected by the UAV at $s_{j,k}$ with duration $t(s_{j,k})$ is

$$V'(s_{j,k}) = \sum_{v \in C(s_j)} \min\{\frac{B \cdot t(s_j)}{K},\ D_v - \frac{B \cdot (k-1) \cdot t(s_j)}{K}\} \mid \frac{D_v}{B} > (k-1) \cdot t(s_j)/K\}. \tag{2.22}$$

It can be seen that $V'(s_{j,1}) \geq V'(s_{j,2}) \geq \ldots \geq V'(s_{j,K})$ because some sensors in $C(s_j)$ have no data for transmissions with the time progress.

Let S' be the set of all virtual potential hovering locations derived from all potential hovering locations, while the latter is obtained the partitioning of the hovering region into a number of squares as we did for the full data collection maximization problem. An auxiliary graph G_s, similar to G_s in the previous subsection, is then constructed. It can be shown that G_s is a metric graph as well, an approximate solution to the corresponding orienteering problem in $G_{S'}$ can be found, by applying `Algorithm` 1, and an approximate solution - a closed tour C' then is delivered. However, tour C' may not be a feasible solution to the partial data collection maximization problem, since it is very likely to have a node $s_{j,k} \in C'$ but $s_{j,k'} \notin C'$ with $k > k'$. Meanwhile, we know that there are no distinctions between node $s_{j,k}$ and $s_{j,k'}$ in terms of their locations and hovering durations as both of them are derived

from the same node s_j. We also know that $V'(s_{j,k}) \leq V'(s_{j,k'})$. Thus, a better solution C'' to the problem can be obtained, by replacing node $s_{j,k}$ in C' with node $s_{j,k'}$. Now, assume that there are $l \geq 2$ nodes from s_j included in tour C', denoted by $s_{j,j_1}, s_{j,j_2}, \ldots, s_{j,j_l}$ with $1 \leq j_l \leq K$ and $1 \leq l \leq K$, they will be replaced by nodes $s_{j,1}, s_{j,2}, \ldots, s_{j,l}$ respectively, and the resulting solution still is a feasible solution to the problem, this node replacement procedure continues until no further replacement is needed. An approximate solution to the problem is obtained in the end. We term this algorithm as `Algorithm 2`.

Algorithm 2 Approximation algorithm for the partial data collection maximization problem without hovering coverage overlapping

Input: An aggregate sensor network $G = (V, E)$ with a set V of aggregate sensor nodes, a UAV with energy capacity $\mathcal{E}$ at depot d, and each node $v \in V$ has data volume D_v, and a given constant $\delta > 0$ but $\delta \leq R_0$.

Output: Find a closed tour including the depot d for the UAV such that the volume of data collected from all aggregate sensor nodes covered by the UAV at hovering locations in the tour is maximized, subject to the energy capacity of the UAV.

1: Partition the monitoring region into M squares $s_1, s_2, \ldots, s_M$ with the edge length of each square being δ; let $S = \{d, s_1, s_2, \ldots, s_M\}$;
2: $S' \leftarrow \cup_{k=1}^{K} \{s_{j,k} \mid s_j \in S\}$;
3: Compute $t(s_{j,k})$, $p(s_{j,k})$, and $w_1(s_{j,k})$ for each $s_{j,k} \in S$ with $1 \leq j \leq M$;
4: Construct an auxiliary graph $G'_s = (S' \cup \{d'\}, E_s \cup \{(v, d') \mid (v, d) \in E_s\}; p(\cdot), w_1(\cdot), w_2(\cdot, \cdot))$, where d' is a dummy depot;
5: Find a simple path P in G'_s between the depot d and the dumpy depot d' such that the total award collected in the path is maximized (as there is no coverage overlapping between any two hovering locations by the assumption), subject to the energy capacity $\mathcal{E}$ of the UAV, by the approximation algorithm for the orienteering problem in metric graphs [8];
6: **for** each j from 1 to M **do**
7: **if** $\exists k, k'$ *s.t.* $k > k', s_{j,k} \in P, s_{j,k'} \notin P$ **then**
8: replace node $s_{j,k}$ by node $s_{j,k'}$
9: **return** A closed tour C derived from P for the UAV, which contains the hovering locations and the sojourn duration at each of the hovering locations.

2.4.4 Algorithm analysis

In the following we show the correctness and time complexity of the proposed algorithms. We first show that the auxiliary graph G_s is a metric graph, as the given approximation algorithm for the orienteering problem is only applicable to metric graphs, we then analyze the time complexity of the two proposed approximation algorithms and their approximation ratios. Notice that both `Algorithm 1` and `Algorithm 2` are approximation algorithms under the assumption that there is no distinction among different hovering locations in each square; otherwise, the problems are intractable due to infinite numbers of potential hovering locations even for a small square area.

Lemma 1. The auxiliary graph G_s is a metric graph.

Proof. Since there is an edge for each pair of nodes in G_s, we show that the edge weights in G_s meet the triangle inequality. For the three edges formed by any three nodes s_j, s_k, and s_l in S, we have

$$\begin{aligned}
&w_2(s_j, s_k) + w_2(s_k, s_l) \\
&= (\frac{w_1(s_j) + w_1(s_k)}{2} + \frac{l(s_j, s_k) \cdot \eta_t}{\nu}) + (\frac{w_1(s_k) + w_1(s_l)}{2} + l(s_k, s_l) \cdot \eta_t) \\
&= \frac{w_1(s_j) + w_1(s_l)}{2} + w_1(s_k) + \frac{(l(s_j, s_k) + l(s_k, s_l)) \cdot \eta_t}{\nu} \\
&\geq \frac{w_1(s_j) + w_1(s_l)}{2} + w_1(s_k) + \frac{l(s_j.s_l) \cdot \eta_t}{\nu} \\
&\geq \frac{w_1(s_j) + w_1(s_l)}{2} + \frac{l(s_j, s_l) \cdot \eta_t}{\nu} \\
&= w_2(s_j, s_l). \qquad (2.23)
\end{aligned}$$

Thus, G_s is a metric graph. □

Theorem 2.2. *Given an aggregate sensor network $G(V, E)$ with each node $v \in V$ having a data volume D_v for collection, and a UAV with energy capacity $\mathcal{E}$ and its depot d, assuming that the moving unit of the UAV is measured by a value of $\delta > 0$ and its coverage range at each hovering location is a circle with radius R_0, there is a $\frac{1}{3}$-approximation algorithm,* `Algorithm` *1, for the full data collection maximization problem in G without hovering coverage overlapping, assuming that the distance difference among potential hovering locations within each square is negligible. The algorithm takes $\mathcal{O}(T_{ort}(\frac{\pi \cdot R_0^2}{\delta^2} \cdot |V|, \frac{\pi^2 \cdot R_0^4}{\delta^4} \cdot |V|^2))$ time, where $T_{ort}(|V'|, |E'|)$ is the time complexity of the approximation algorithm of Bansal et al. [8] for the orienteering problem in a graph with $|V'|$ nodes and $|E'|$ edges.*

Proof. We first show that the solution obtained by `Algorithm` 1 is feasible. It can be seen that the closed tour C is a simple closed tour including the depot d. We show that the total energy consumption of the UAV on the closed tour C is no greater than $\mathcal{E}$. As the total length of C is no greater than $\mathcal{E}$, the energy consumption of the UAV on C (hovering at nodes and traveling on edges) is the weighted sum of edges in C, which is no greater than its energy capacity. Furthermore, for each hovering location s_j in C, assume that s_i and s_k are its two neighboring hovering locations in C, then the hovering energy consumption $w(s_j)$ of the UAV at location s_j is distributed to its two incident edges (s_i, s_j) and (s_j, s_k) as part of their weights, i.e., the energy weights of the two edges are $w_2(s_i, s_j) = \frac{w_1(s_i)+w_1(s_j)}{2} + \frac{l(s_i,s_j)\cdot\eta_t}{\nu}$ and $w_2(s_j, s_k) = \frac{w_1(s_j)+w_1(s_k)}{2} + \frac{l(s_j,s_k)\cdot\eta_t}{\nu}$, respectively. Thus, the UAV has sufficient energy at each hovering location s_j to collect all data from the IoT devices in $C(s_j)$.

We then analyze the time complexity of the proposed algorithm, `Algorithm` 1. We notice that the number of squares, M, is a linear function of the number of aggregate sensor nodes in V. For example, assume that the coverage range of the UAV at

a hovering location is a circle with radius R_0, then, the number of its potential hovering locations for collecting data from an IoT device $v \in V$ will be no greater than $\lceil \frac{\pi \cdot R_0^2}{\delta^2} \rceil$ in terms of the number of squares covering v. Thus, the maximum number of squares in G_s is no greater than $\sum_{v \in V} \lceil \frac{\pi \cdot R_0^2}{\delta^2} \rceil \leq (\frac{\pi R_0^2}{\delta^2} + 1) \cdot |V|$ as both R_0 and δ usually are constants. There exists an edge between every pair of the node in G_s, therefore, G_s contains $|E_s| = \mathcal{O}(|S|^2) = \mathcal{O}(\frac{\pi^2 \cdot R_0^4}{\delta^4} \cdot |V|^2)$ edges. Finding a $\frac{1}{3}$-approximate solution (a closed tour C) for the orienteering problem in G_s starting at node d takes $\mathcal{O}(T_{ort}(\frac{\pi \cdot R_0^2}{\delta^2} \cdot |V|, \ \frac{\pi^2 \cdot R_0^4}{\delta^4} \cdot |V|^2))$ time, by the approximation algorithm due to Bansal *et al.* [8], assuming that the distances of hovering locations within each square are negligible, where $T_{ort}(|V'|, |E'|)$ is the time complexity of the approximation algorithm of Bansal *et al.* [8] for the orienteering problem in a graph with $|V'|$ nodes and $|E'|$ edges. □

We finally show that the solution delivered by `Algorithm 2` for the partial data collection maximization problem is feasible, and analyze its approximation ratio as follows.

Theorem 2.3. *Given an aggregate sensor network $G(V, E)$ with each node $v \in V$ having a data volume D_v for collection, and a UAV with energy capacity $\mathcal{E}$ and its depot d, assuming that the moving unit of the UAV is measured by a value of $\delta > 0$ and its coverage range at each hovering location is a circle with radius R_0, there is a $\frac{1}{3}$-approximation algorithm,* `Algorithm 2`*, for the partial data collection maximization problem in G without hovering coverage overlapping, assuming that the distance difference among potential hovering locations within each square is negligible. The algorithm takes $\mathcal{O}(T_{ort}(\frac{\pi \cdot R_0^2}{\delta^2} \cdot |K \cdot V|, \ \frac{\pi^2 \cdot R_0^4}{\delta^4} \cdot |K \cdot V|^2))$ time, where $T_{ort}(|V'|, \ |E'|)$ is the time complexity of the approximation algorithm of Bansal et al. [8] for the orienteering problem in a graph with $|V'|$ nodes and $|E'|$ edges.*

Proof. The proof of Theorem 2.3 is almost identical to the one for Theorem 2.2, and it can be shown that the auxiliary graph G_s is also a metric graph. The only difference lies in the analysis of the approximation ratio of the approximation algorithm. Assume that there is an optimal solution for the partial data collection maximization problem OPT_K, as the solution A_K delivered by the approximation algorithm is $OPT_K/3$, while a feasible solution to the problem is a solution by replacing all l virtual nodes derived from a hovering location node with its first l virtual nodes, and the accumulative volume of the removed virtual nodes is no more than the accumulative volume of the l added virtual nodes, and the total volume of data collected in the resulting tour is no less than that of the initial solution. Thus, the solution is no less than one third of the optimal one.

The proof of time complexity is similar to the one in the proof body of Theorem 2.3, omitted. □

2.5 Heuristic algorithm for the data collection maximization problems

In this section, we deal with the full and partial data collection maximization problems with hovering coverage overlapping, by proposing efficient heuristic algorithms for them. We also conduct the computing complexity analysis of the proposed algorithms.

2.5.1 Algorithm for the full data collection maximization problem with hovering coverage overlapping

The basic idea behind the proposed algorithm is to find a closed tour for the UAV iteratively. Initially, the closed tour consists of only the depot. Within each iteration, a new hovering location is added to the tour. For the sake of convenience, we assume that a partially closed tour that consists of hovering locations $s_0, s_1, \ldots, s_{j-1}$ has been constructed. Let $S_{j-1} = \{s_0, s_1, s_2, \ldots, s_{j-1}\}$ be the set of chosen hovering locations for the UAV so far and $s_0 = d$, which implies that the sum of energy consumptions on these j hovering locations and traveling along the closed tour $TSP(S_{j-1})$ is no more than $\mathcal{E}$, where $TSP(S_{j-1})$ is the length of the closed tour induced by the nodes in set S_{j-1}, which is obtained by applying Christofides's algorithm for the Travelling Salesman Problem [26]. Recall that the coordinates of location s_j are (x_j, y_j, H), and $C(s_j) = \{v_i \mid v_i \in V \ \& \ \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_0\}$ is the set of IoT devices covered by the UAV at hovering location s_j, i.e., each IoT device in $C(s_j)$ is able to transmit its data to the UAV located at s_j. The rest is to determine the next hovering location s_j as follows.

The volume of data collected by the UAV when it is located at s_j is

$$V'(s_j) = \sum\{D_v \mid v \in C(s_j) \setminus \cup_{j'=0}^{j-1} C(s_{j'})\}, \tag{2.24}$$

i.e., if the stored data of an aggregate sensor node has been fully collected in any of the previous $j-1$ hovering locations, then the aggregate sensor node will not contribute any award to the optimization objective.

The hovering duration of the UAV for data collection at hovering location s_j is

$$t'(s_j) = \max_{s_j \in S \setminus S_{j-1}} \{\frac{D_v}{B} \mid v \in C(s_j) \setminus \cup_{l=0}^{j-1} C(s_l)\}. \tag{2.25}$$

Denote by the ratio $\rho(s_j)$ of the volume of data collected to the total amount of energy consumed by the UAV on hovering and traveling to and from location s_j as

follows.

$$\rho(s_j) = \frac{V'(s_j) \cdot \nu}{t'(s_j) \cdot \eta_h \cdot \nu + (TSP(S_j) - TSP(S_{j-1})) \cdot \eta_t}$$
$$\text{if} \sum_{s_{j'} \in S_{j-1} \cup \{s_j\}} t'(s_{j'}) \cdot \eta_h + \frac{TSP(S_j) \cdot \eta_t}{\nu} \leq \mathcal{E}, \tag{2.26}$$

where $S_j = S_{j-1} \cup \{s_j\}$. Notice that $\sum_{s_{j'} \in S_{j-1}} t'(s_{j'}) \cdot \eta_h + \frac{TSP(S_{j-1}) \cdot \eta_t}{\nu} \leq \mathcal{E}$ holds, as S_{j-1} is a feasible solution to the problem, following the assumption. The hovering location s_j is chosen as the next hovering location of the UAV if its ratio $\rho(s_j)$ is the maximum one among all potential hovering locations in $S \setminus S_{j-1}$, and the total energy consumption of the UAV in the closed tour including s_j is no greater than its energy capacity. This procedure continues until no more hovering locations can be added to the closed tour without violating the energy capacity of the UAV.

The detailed algorithm for the full data collection maximization problem with hovering coverage overlapping is given in `Algorithm` 3.

Algorithm 3 A heuristic algorithm for the full data collection maximization problem with hovering coverage overlapping

Input: An aggregate sensor network $G = (V, E)$ with a set V of aggregate sensor nodes, a UAV with energy capacity $\mathcal{E}$ at depot d, and each node $v \in V$ has data volume D_v, and a given constant $\delta > 0$.

Output: Find a closed tour including depot d for the UAV such that the volume of data collected from all aggregate sensor nodes covered by it at its hovering locations in the tour is maximized, subject to the energy capacity of the UAV.

1: Partition the monitoring region into M squares $s_0, s_1, \ldots, s_M$, let $S = \{s_0, s_1, \ldots, s_M, d'\}$;
2: Construct the closed tour for the UAV iteratively, $S_0 = \{d\}$ initially;
3: $j \leftarrow 1$;
4: **while** $\sum_{s_{j'} \in S_{j-1}} t'(s_{j'}) \cdot \eta_h + \frac{TSP(S_{j-1}) \cdot \eta_t}{\nu} < \mathcal{E}$ **do**
5: Choose the next hovering location $s_j \in S \setminus S_{j-1}$ such that the ratio $\rho(s_j)$ is the maximum one, i.e., $s_j = \text{argmax}\{\frac{V'(s_{j'}) \cdot \nu}{t'(s_{j'}) \cdot \eta_h \cdot \nu + (TSP(S_j) - TSP(S_{j-1})) \cdot \eta_t} \mid$
6: $s_{j'} \in S \setminus S_{j-1}$, $\sum_{s_{j'} \in S_{j-1} \cup \{s_j\}} t'(s_{j'}) \cdot \eta_h + \frac{TSP(S_j) \cdot \eta_t}{\nu} \leq \mathcal{E}\}$;
7: $S_j \leftarrow S_{j-1} \cup \{s_j\}$;
8: $j \leftarrow j + 1$;
9: **return** the closed tour with the hovering location sequence $s_0, s_1, \ldots, s_j$.

2.5.2 Algorithm for the partial data collection maximization problem with hovering coverage overlapping

In the following, we consider the partial data collection maximization problem with hovering coverage overlapping, by adopting the similar technique for the full

data collection maximization problem with hovering coverage overlapping. We show how to modify `Algorithm 3` for this purpose. Specifically, we treat each virtual hovering location derived from each (real) potential hovering location as a potential hovering location of the UAV as we did for `Algorithm 3`. However, we only allow one virtual hovering location of each potential hovering location to be included in the closed tour, as the tour must be a simple closed tour. If two virtual hovering locations s_{j,k_1} and s_{j,k_2} derived from the same hovering location s_k are chosen to be included in the closed tour with $1 \leq k_1 < k_2 \leq K$, then only location s_{j,k_2} will be included while location s_{j,k_1} will be removed from the tour. Notice that all data that are supposed to be collected by the UAV at hovering location s_{j,k_1} for sojourn duration $t'(s_{j,k_1})$ will be collected by the UAV at location s_{j,k_2}, since the UAV at s_{j,k_2} takes a longer sojourn duration $t'(s_{j,k_2})$ for data collection while $t'(s_{j,k_2}) > t'(s_{j,k_1})$. It is also noted that the volume of data in a sensor $v \in V$ may be collected by the UAV at multiple hovering locations if the stored data in v has not been fully collected yet. We here use an example to illustrate this scenario. Assume that the duration of collecting all data from a sensor v is $t(v) = D_v/B$ time units. We further assume that v is covered by the UAV at three different hovering locations s_{j_1}, s_{j_2} and s_{j_3} with sojourn time t_1, t_2 and t_3, respectively, i.e., $v \in C(s_{j_1})$, $v \in C(s_{j_2})$, and $v \in C(s_{j_3})$. If $t(v) \geq t'(s_{j_1}) + t'(s_{j_2}) + t'(s_{j_3}) = t_1 + t_2 + t_3$, then the rest of data stored at sensor v can still be collected by the UAV at these three hovering locations. Therefore, the residual data volume and the hovering duration at some virtual hovering locations must be recalculated after a virtual hovering location s_j is added to the closed tour, since the data of some aggregate sensor nodes in $C(s_j)$ are contained by these potential virtual hovering locations and their data have been partially collected in the previous hovering locations already, where node $s_{j,k}$ is defined as follows.

$$s_{j,k} = \text{argmax}\{\frac{V'(s_{j',k}) \cdot \nu}{t'(s_{j',k}) \cdot \eta_h \cdot \nu + (TSP(S'_j) - TSP(S'_{j-1})) \cdot \eta_t} \mid s_{j',k} \in S' \setminus S'_{j-1},$$
$$\sum_{s_{j',k} \in S'_{j-1} \cup \{s_{j',k}\}} t'(s_{j',k}) \cdot \eta_h + \frac{TSP(S_j) \cdot \eta_t}{\nu} < \mathcal{E}\}. \quad (2.27)$$

The detailed algorithm for the full data collection maximization problem with hovering coverage overlapping is given in `Algorithm 4`.

2.5.3 Analysis of the proposed algorithm

The rest is dedicated to the correctness proof and time complexity analysis of the proposed algorithms. In the following we first analyze the time complexity of the proposed algorithm, `Algorithm 3`.

Theorem 2.4. *Given an aggregate sensor network $G(V, E)$ with each node $v \in V$ having a data volume D_v for collection, and a UAV with energy capacity $\mathcal{E}$ and depot d, assuming that the moving unit of the UAV is measured by $\delta > 0$ and its hovering coverage range of the UAV*

Algorithm 4 A heuristic algorithm for the partial data collection maximization problem with hovering coverage overlapping

Input: A sensor network $G = (V, E)$ with a set V of aggregate sensor nodes, a UAV with energy capacity $\mathcal{E}$ at depot d, and each node $v \in V$ has data volume D_v, and a given constant $\delta > 0$.

Output: Find a closed tour for the UAV including depot d such that the volume of data collected from aggregate sensor nodes within the hovering locations in the tour is maximized, subject to the energy capacity of the UAV.

Partition the monitoring region into M squares $s_0, s_1, \ldots, s_M$;
$S' \leftarrow \cup_{k=1}^{K} \{s_{j,k} \mid s_j \in S\}$;
Construct the closed tour for the UAV, $S'_0 = \{d\}$ initially;
$j \leftarrow 1$;
while $\sum_{s'_j \in S'_{j-1}} t'(s_{j'}) \cdot \eta_h + \frac{TSP(S'_{j-1}) \cdot \eta_t}{\nu} < \mathcal{E}$ **do**
    Choose a location $s_{j,k} \in S' \setminus S'_{i-1}$ by Eq. (2.27) such that the ratio $\rho(s_{j,k})$ is the maximum one;
    $S'_j \leftarrow S'_{j-1} \cup \{s_{j,k}\} \setminus \{s_{j,k'} \mid 1 \leq k' < k\}$;
    $S' \leftarrow S' \setminus \{s_{j,k'} \mid 1 \leq k' \leq k\}$; /* as only one virtual hovering location from each hovering location is added to the tour */
    if $\exists k' : s_{j,k'} \in S'_{k-1}$ with $k' < k$ **then**
        $S'_j \leftarrow S'_{j-1} \setminus \{s_{j,k'}\}$;
        Calculate the data volume $D_v^{(j)}$ of each sensor $v \in C(s_{j,k})$;
        Calculate $V'(s_{j',k'})$ and $t'(s_{j',k'})$ for each potential location $s_{j',k'} \in S' \setminus S'_j$ if $C(s_{j',k'}) \cap C(s_{j,k}) \neq \emptyset$;
    $j \leftarrow j + 1$;
return the closed tour that consists of the hovering location sequence $s_0, s_1, \ldots, s_j$ can be derived from S'_j,

at each hovering location is a circle with radius R_0, there is an efficient heuristic algorithm, `Algorithm` *3, for the full data collection maximization problem in G with hovering coverage overlapping. The algorithm takes $\mathcal{O}(\frac{\pi^4 \cdot R_0^8}{\delta^8} \cdot |V|^4)$ time.*

Proof. `Algorithm` 3 proceeds iteratively, and the number of iterations is bounded by $|S| = M$. Within iteration j with $1 \leq j \leq M$, identifying the next hovering location s_j is performed through the calculation of $\rho(s_j)$ for every location $s_j \in S \setminus S_{j-1}$. This takes $\mathcal{O}(|S \setminus S_{j-1}| \cdot |V| + |S_j|^3) = \mathcal{O}(M \cdot |V| + M^3)$ time, due to the fact that the calculation of $TSP(S_j)$ takes $\mathcal{O}(|S_j|^3)$ time by Christofides' algorithm [26]. The proposed algorithm thus takes $\mathcal{O}(M^2 \cdot |V| + M^4) = \mathcal{O}(\frac{\pi^4 \cdot R_0^8}{\delta^8} \cdot |V|^4)$ time, since we proved that $M \leq (\frac{\pi R_0^2}{\delta^2} + 1) \cdot |V|$ in Section 2.4.4. Notice that in practice, the values of both δ and R_0 are constants, the time complexity of `Algorithm` 3 thus is $\mathcal{O}(|V|^4)$. □

We then analyze the correctness of `Algorithm` 4. To this end, we show that the amount of data that is supposed to be collected by the UAV at s_{j,k_1} will be collected by the UAV at s_{j,k_2} if $k_1 < k_2$ by the following lemma.

Lemma 2. For a given hovering location s_j, if one of its virtual hovering locations s_{j,k_1} is included in the closed tour and its another virtual hovering location s_{j,k_2} is chosen to be added to the tour later with $k_2 > k_1$, then s_{j,k_1} will be removed from the closed tour by the proposed algorithm, `Algorithm` 4. By doing so will not reduce the amount of data collected when the UAV at hovering location s_j, and the correctness of the proposed algorithm holds.

Proof. We note that for each potential hovering location, at most one of its virtual hovering locations is included in the closed tour. Following `Algorithm` 4, if at most one virtual hovering location derived from a potential hovering location in S is included in the closed tour, the solution is feasible. Otherwise, assume that a virtual hovering location s_{j,k_1} of $s_j \in S$ at iteration i_1 of the algorithm has been added to the closed tour already, another virtual location s_{j,k_2} of s_j at iteration i_2 is added to the closed tour again, where $1 \leq i_1 < i_2 \leq |S|$. To ensure that at most one virtual hovering location for each hovering location is added to the closed tour, the hovering location s_{j,k_1} will be removed while s_{j,k_2} is added to the tour, and all the other virtual hovering locations $s_{j,k'}$ of s_j with $k' < k_2$ will be removed from S for further consideration by the algorithm. Despite that the volume of data collected and sojourn durations at each hovering location from iterations $i_1 + 1$ to $i_2 - 1$ could be changed due to the removal of s_{j,k_1} from the closed tour, we will not update these hovering locations and their sojourn durations as the residual volume of data at each IoT device in $C(s_{j,k_1})$ (iteration i_1) in fact is larger than when the virtual hovering location s_{j,k_1} is included in the tour. The volume of all data that was supposed to be collected by the UAV at location s_{j,k_1} with sojourn duration $k_1 \cdot t(s_j)/K$ will be later collected by the UAV at location s_{j,k_2} (iteration i_2) with sojourn duration $k_2 \cdot t(s_j)/K$. Thus, the total volume of data collected from the closed tour by the removal of s_{j,k_1} does not change. □

We then have the following theorem.

Theorem 2.5. *Given an aggregate sensor network $G(V, E)$ with each node $v \in V$ having a data volume D_v for collection, and a UAV with energy capacity $\mathcal{E}$ and depot d, assuming that the moving unit of the UAV is measured by $\delta > 0$ and its hovering coverage range of the UAV at each hovering location is a circle with radius R_0, there is a heuristic algorithm,* `Algorithm` *4, for the partial data collection maximization problem with hovering coverage overlapping, assuming that the distance differences among the potential hovering locations within each square are negligible. The algorithm takes $\mathcal{O}(\frac{\pi^4 \cdot R_0^8}{\delta^8} \cdot K^4 \cdot |V|^4)$ time, where K is a given integer with $K \geq 1$.*

Proof. The correctness of the solution delivered by `Algorithm` 4 is shown by Lemma 2, omitted. The time complexity analysis of `Algorithm` 4 is similar to the one in the proof body of Theorem 2.4. The only difference between the two algorithms lies in the fact that we now have $M' = K \cdot M$ virtual squares instead of M squares in Theorem 2.4, omitted. □

2.6 Performance evaluation

In this section, we evaluate the performance of the proposed algorithms for the full and partial data collection maximization problems through experimental simulations. We also investigate the impact of important parameters on the algorithm performance.

Table 2.1: Table of Parameter Settings

Parameters	Values
Sensing field	1,000 m × 1,000 m
Network size	500 aggregate sensor nodes
Node data volume D_v	100 MB - 1,000 MB
Transmission range R	70 m
Flying altitude H	50 m
Data rate B	150 $Mbps$
Energy capacity $\mathcal{E}$	$3 \times 10^5 joules$
Flying speed nu	1
Traveling consumption rate η_t	100 J/s
Hovering energy rate η_h	150 J/s

2.6.1 Experimental Settings

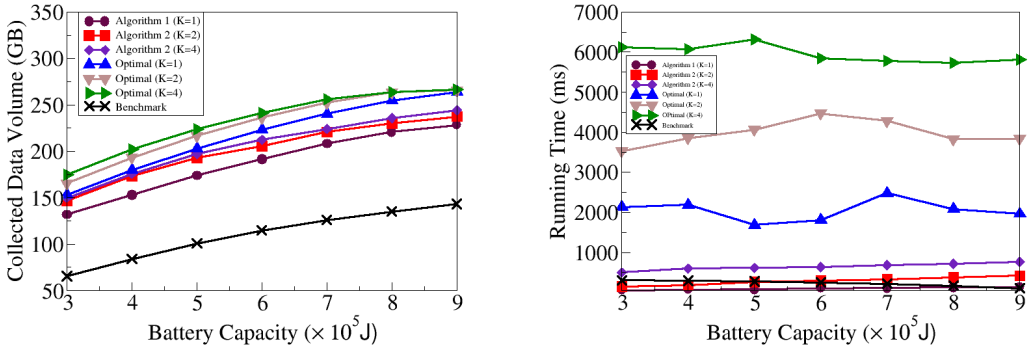
We consider a sparse sensor network that consists of 500 aggregate sensor nodes randomly deployed in a 1,000 *meters* × 1,000 *meters* square. The data volume of each aggregate sensor node is randomly drawn from 100 MB to 1,000 MB. Without loss of generality, we assume that the transmission range R of each aggregate sensor node is 70 *meters* and the data transmission rate of the sensor is 150 $Mbps$ [122]. We assume that a UAV is deployed at a depot d initially. The UAV has energy capacity $\mathcal{E} = 3 \times 10^5$ *joules* at constant flying speed $v = 10$ m/s and the flying altitude

$H = 50$ *meters*. The energy consumption rates of the UAV on traveling and hovering are $\eta_t = 100$ J/s and $\eta_h = 150$ J/s, respectively [102]. The value in each figure is the mean of the results out of 50 network instances of the same size. The running time of an algorithm is obtained based on a machine with 3.6 GHz Intel i7 single-core CPU and 16 GB RAM. Unless otherwise specified, these parameters will be adopted in the default setting. Table 2.1 lists the default settings of the parameters in this chapter.

To evaluate the performance of the proposed algorithms, we here introduce an iterative heuristic `benchmark` as follows. It starts finding a closed tour C including all aggregate sensor nodes and the depot by the Christofides' algorithm. If the total amount of energy consumed in C is no greater than the energy capacity of a UAV, done. Otherwise, a node in the tour is chosen if its removal will result in the minimum loss of data volume per unit energy consumption. This procedure continues until the total energy consumption of the resulting closed tour is no greater than $\mathcal{E}$.

2.6.2 Performance evaluation of different algorithms for the full and partial data collection maximization problems without hovering coverage overlapping

We first investigate the performance of different algorithms for the full data collection maximization problems without hovering coverage overlapping. As shown in Fig. 2.3(a), `Algorithm` 1 outperforms the benchmark algorithm, and collects at least 80% of the data in the optimal solution. For example, when $\mathcal{E} = 3 \times 10^5$ *Joules*, the volume of data collected by `Algorithm` 1 is around twice the amount by the benchmark algorithm, and the volume gap between them becomes larger with the increase on the energy capacity of the UAV. Fig. 2.3(b) plots the running time curves of the two mentioned algorithms and ILP solver, from which it can be seen that the running time of `Algorithm` 1 is much shorter than ILP solver.

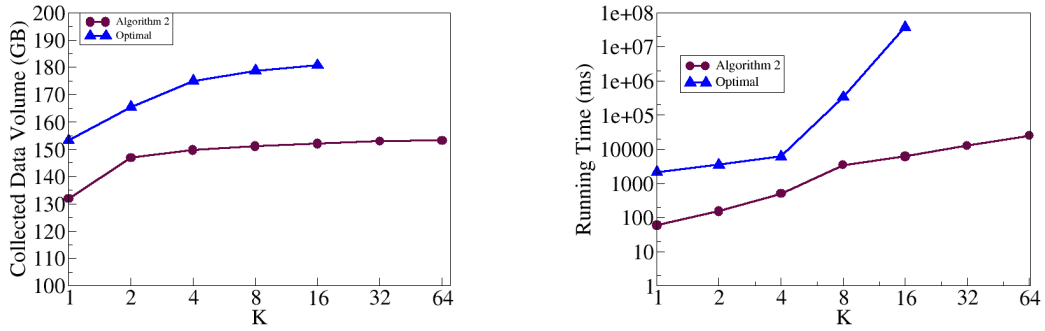


(a) Collected data volume (b) The running time

Figure 2.3: Performance of different algorithms for the full data collection maximization problems without hovering coverage overlapping by varying the energy capacity of the UAV.

We also study the performance of the proposed algorithm for the partial data collection maximization problem. Fig. 2.4(a) shows that with the increase on K, the

UAV collects more data per tour. For instance, when $\mathcal{E} = 3 \times 10^5$ *Joules*, the volume of data collected by `Algorithm` 2 when $K = 4$ is 149.8 *GB*, while the volume of data collected by `Algorithm` 1 is 131.9 *GB*. However, the growth of K results in a longer running time of `Algorithm`2 and ILP solver . E.g., when $K = 1$, the running time for `Algorithm` 2 and ILP solver is around $100ms$ and $2,000ms$ respectively, but when $K = 16$, the time consuming is around $24,830$ *ms* and 4×10^7 *ms* respectively. When $K \geq 32$, ILP solver cannot obtain the result within reasonable time.



(a) Collected data volume

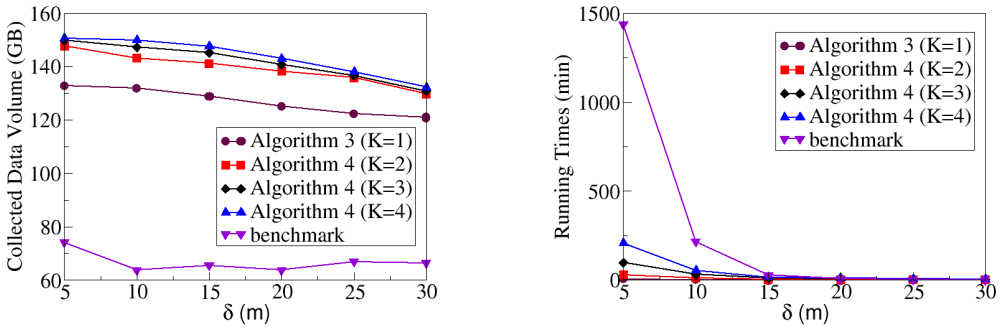
(b) The running time

Figure 2.4: Performance of different algorithms for the data collection maximization problem without hovering coverage overlapping.

2.6.3 Performance evaluation of algorithms for the full and partial data collection maximization problems with hovering coverage overlapping

We then study the performance of `Algorithm` 3 and `Algorithm` 4 for the full data collection maximization problem with hovering coverage overlapping against the benchmark algorithm. It can be seen from Fig.2.5(a) that `Algorithm` 3 and `Algorithm` 4 outperform the benchmark algorithm significantly. Furthermore, `Algorithm` 4 is superior to `Algorithm` 3 as the latter is a special case of the former when $K = 1$. When $\delta = 5$ meters, the collected data volume by `Algorithm` 3 and `Algorithm` 4 ($K = 2$) are 132.8 *GB* and 147.7 *GB*, respectively, which are 79.1% and 99% higher than that of the solution delivered by the benchmark algorithm (74.14 *GB*).

We also study the performance of `Algorithm` 4 by varying the value of K. It can be seen from Fig. 2.5(a) that a larger K will result in more data collected per tour, this is due to more accurate planning for data collection per unit energy consumption. For instance, the collected data volume increases from 147.7 *GB* to 150.7 *GB* when K increases from 2 to 4. Fig. 2.5(b) indicates that `Algorithm` 4 takes a longer running time than that of `Algorithm` 3 with the growth of K. For example, the running time of `Algorithm` 4 is about 54.1 minutes when $K = 4$, which is around 50 times of the running time 1.61 minutes of `Algorithm` 3.



(a) Collected data volume

(b) The running time

Figure 2.5: Performance of different algorithms for the full and partial data collection maximization problems with hovering coverage overlapping, by varying the value of δ from 5 meters to 30 meters when $|V| = 500$.

2.6.4 Impact of important parameters on the performance of the proposed algorithms for the full and partial data collection maximization problems with and without hovering coverage overlapping

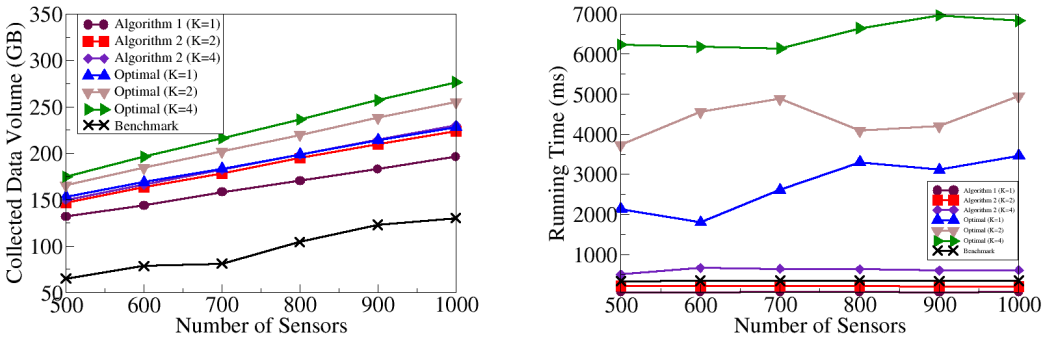
In the following we evaluate the impact of parameters $\delta, \mathcal{E}, |V|$ on the performance of the proposed algorithms for the full and partial data collection maximization problems without hovering coverage overlapping.

We start with the impact of the parameters on the performance of the proposed algorithms for the full data collection maximization problem without hovering coverage overlapping.

(a) We analyze the impact of energy capacity $\mathcal{E}$ of the UAV, by varying it from 3×10^5 to 9×10^5 *Joules* while fixing other parameters. Fig. 2.3(a) shows that the volume of data collected by all algorithms increases with the growth of energy capacity. When $\mathcal{E} = 9 \times 10^5$, `Algorithm` 1 collects 73.09%($\approx (228.3 - 131.9)/131.9$) more data than the one when $\mathcal{E} = 3 \times 10^5$. A larger battery capacity implies that the UAV can visit more hovering locations and hovering longer at hovering locations, hence increasing the running time of the algorithm. On the other hand, from Fig.2.3(b), it can be seen that the running time of `Algorithm` 1 and `Algorithm` 2 increases with the growth of the UAV energy capacity, because more hovering locations needs planning. While the running time of the benchmark algorithm decreases with the growth of the UAV energy capacity, because fewer nodes will be pruned from the initial TSP tour as more energy can be consumed in the tour.

(b) We study the impact of the number of aggregate sensor nodes $|V|$ on the performance of `Algorithm` 1 and `Algorithm` 2. Fig. 2.6(a) depicts the data collections of the UAV while varying the number of aggregate sensor nodes in the monitoring region from 500 to 1,000, which shows the amount of collected data goes up when there are more aggregate sensor nodes. When there are 1,000 nodes, `Algorithm` 1 collects 196.3 GB, which is approximately 64.4GB higher than that when there are 500 nodes. Fig. 2.6(b) plots the running time curve of different algorithms. It can be

seen that `Algorithm` 1 and `Algorithm` 2 remain stable when the number of nodes increases from 500 to 1000. The reason is that the hovering sojourn duration at each hovering location is determined by the sensor within the coverage range of the UAV with the maximum data volume.



(a) Collected data volume

(b) The running time

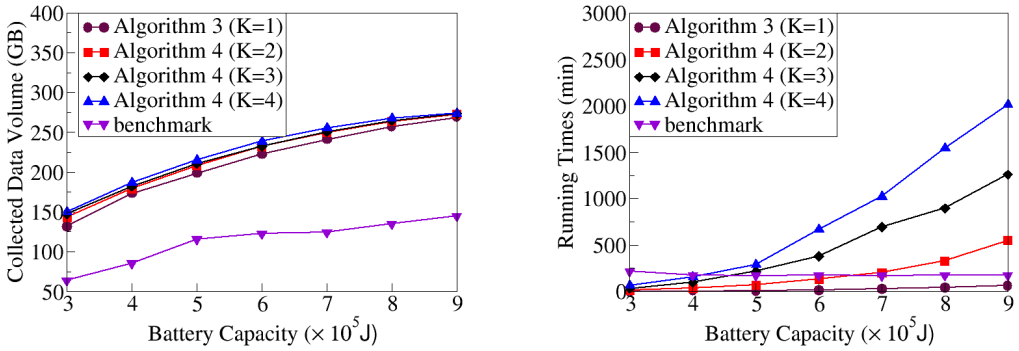
Figure 2.6: Performance of different algorithms for the data collection maximization problem without hovering coverage overlapping by varying the number of sensors from 500 to 1,000.

We now evaluate the impact of parameters δ, $\mathcal{E}$ and $|V|$ on the performance of `Algorithm` 3 and `Algorithm` 4 for the full and partial data collection maximization problems with hovering coverage overlapping.

(a) We evaluate the impact of δ on the performance of the two mentioned algorithms. It can be seen from Fig. 2.5(a) that for a fixed $K \geq 1$ of number of partitionings on the sojourn duration at each hovering location, the total volume of data collected per tour reduces, so do the running times of the algorithms because the number of potential hovering locations for the UAV becomes smaller, and less data will be collected. E.g., when $K = 4$, it can be seen from Fig. 2.5(a) that when $\delta = 5$ meters, the collected data volume is about 13.9% higher than that by itself when $\delta = 30$ meters. Therefore, when δ is sufficiently small, the total volume of data collected by the UAV is maximized. Fig. 2.5(b) depicts the running times of the two mentioned algorithms.

(b) We investigate the impact of the energy capacity $\mathcal{E}$ of the UAV on the performance of different algorithms, by varying it from 3×10^5 *joules* to 9×10^5 *joules* while fixing the value of δ at 10 meters. Fig. 2.7(a) illustrates that the collected data volume goes up with the increase on the energy capacity of the UAV, since the UAV can visit more hovering locations with longer hovering durations to collect more data from its hovering locations. For example, when $K = 4$, the collected data is increased by 82% with the growth of the energy capacity of the UAV from 3×10^5 *joules* to 9×10^5 *joules*. Fig. 2.7(b) demonstrates the impact of the battery capacity of UAV on the running time of the algorithm. A larger battery capacity implies that `Algorithm` 3 and `Algorithm` 4 can visit more hovering locations, hence increasing their running time. On the other hand, with more energy capacity, the benchmark algorithm will remove fewer nodes from the initial closed tour $\mathcal{C}$, which leads to a shorter running time.

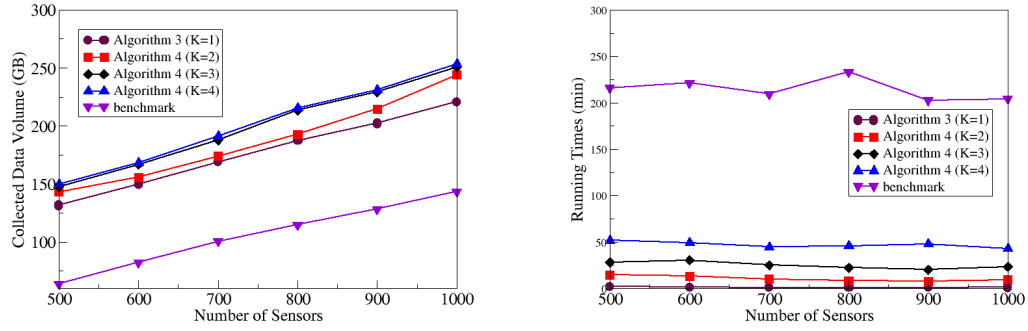
`Algorithm` 4 thus needs more running time compared with that of the benchmark algorithm with the growth of the energy capacity on the UAV.



(a) Collected data volume (b) The running time

Figure 2.7: Performance of different algorithms, by varying the battery capacity of the UAV from 3×10^5 *joules* to 9×10^5 *joules*.

(c) We study the impact of aggregate sensor nodes $|V|$ on the performance of `Algorithm` 3 and `Algorithm` 4, by varying the number of aggregate sensor nodes from 500 to 1,000. Fig. 2.8(a) shows that volume of data collected grows with the increase on the number of aggregate sensor nodes. On the other hand, Fig. 2.8(b) depicts that the running time of each comparison algorithm does not change too much, by increasing the number of sensors. The rationale behind is that the hovering sojourn duration at each hovering location is determined by the sensor with the maximum data volume, due to the OFDMA technique.



(a) Collected data volume (b) The running time

Figure 2.8: Performance of different algorithms by varying the number of sensors from 500 to 1,000.

2.7 Conclusions

In this chapter, we studied the full or partial data collection maximization problem for IoT applications, using an energy-constrained UAV. We first proposed a novel data collection framework that enables the UAV to collect sensory data from multiple IoT devices simultaneously. We then formulated the full and partial data collection maximization problems that allow the UAV to fully or partially collect data from IoT devices at each of its hovering locations. We thirdly showed that both problems are NP-hard, and instead devised efficient approximation and heuristic algorithms for the problems. We finally evaluated the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising.

Chapter 3

Digital Twin Synchronizations and Fidelity-Aware Query Services

In this chapter, we first formulate a novel staleness minimization problem of DT states with the aim to minimize the staleness of DT states of sensors. We then devise a heuristic algorithm for the staleness minimization problem with the assumption that the data volume of each sensor since its last data collection is given. To remove this assumption, we also develop a deep learning mechanism to predict the volume of data generated by each sensor. We then make use of the prediction results to determine the next UAV tour. Built upon the DT state information of sensors, we consider fidelity-aware queries on the DT data by developing a cost-effective evaluation plan, where the fidelity of a query result is referred to as data freshness of the query result.

3.1 Introduction

To minimize the average staleness of DT states of all sensors for the given time horizon, there is a non-trivial trade-off between the freshness of collected data (the Age of Information) and the synchronization (the DT state update) cost per update round. The freshness of DT states can be improved by increasing the update budget per update round, e.g., dispatching multiple instead of a single UAV to collect data from sensors. Since the energy capacity on each UAV is limited, a UAV can only collect data from some but not all sensors at each tour, this implies that different sensors may have different synchronization time points with their DTs, i.e., the staleness of DT states of different sensors are different. Minimizing the average staleness of DT states of sensors for a given time horizon subject to the update budget per update round is challenging, as the synchronization cost of the DT of a sensor is determined not only by the volume of data generated of the sensor but also by the distances of the sensor from the rest of sensors whose data to be collected in that UAV tour.

The novelty of this chapter lies in addressing the synchronization issue on digital twins with their physical objects (sensors) under a limited synchronization budget at each update round, through a real-world example - a UAV-enabled data collection for a remote sensor network. We explore non-trivial interactions (synchronizations) between physical objects (sensors) and their digital twins. That is, how to update

the DT states of sensors such that the average staleness of the DT states of all sensors can be minimized under the limited synchronization budget for a finite time horizon? and how to make use of the DT state information of sensors to accurately predict the behaviours of their physical objects (sensors)? The predicted volume of data generated by each sensor since its last synchronization can then be used to determine the updating of DT states of which sensors in the next update round. Built upon the data collected from sensors stored in their DTs, we consider fidelity-aware queries and develop a cost-effective evaluation plan for such queries in the MEC network. To the best of our knowledge, this is the first synchronization framework of budget-constrained DT state synchronizations between DTs and their physical objects, and the synchronization algorithms for the proposed framework have also been proposed.

The rest of this chapter is organized as follows. Section 3.2 introduces notions, notations, and problem definitions. Section 3.3 provides an optimal solution to a special case of the staleness minimization problem, and Section 3.4 devises an efficient algorithm for the problem. Built upon the DT information, Section 3.5 deals with fidelity-aware queries on DT information by developing cost-effective evaluation plans for the queries. Section 3.6 evaluates the performance of the proposed algorithms, and Section 3.7 concludes the chapter.

3.2 Preliminaries

In this section we first introduce the system model, notions and notations. We then define problems and show NP-hardness of the two defined problems.

3.2.1 System model

Given a mobile edge computing (MEC) network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, $\mathcal{N}$ is the set of Access Points (APs) and $\mathcal{E}$ is the set of links between APs. With each AP, there is a co-located cloudlet that is connected with the AP through a high speed optical fiber cable, and the communication delay between the AP and its co-located cloudlet can be ignored. An AP and its colocated cloudlet can be used interchangeably if no confusion arises. The digital twins (DTs) of different physical objects are stored in cloudlets. The states of DTs can be synchronized (updated) with their physical objects at different update rounds. Users can make use of services provided by the MEC service provider through issuing queries on the DT data stored at cloudlets in the MEC network.

In this chapter, we study a general setting of DT state synchronizations with their physical objects through a real application scenario, where there is a renewable sensor network deployed in a remote region. Let V be the set of sensors in the sensor network. Each sensor $v_i \in V$ has a digital twin DT_i in a cloudlet in an MEC network $\mathcal{G}$, which is a mirror or a modelling of the sensor and contains all data collected from sensor v_i in the past, including the volume of data collected at each of previous update time slots. We term all collected data related to the DT of a sensor to its last

update as the *DT state* of the sensor. We further assume that DT_i of sensor v_i stores all information related to sensor v_i so far, e.g., at which time point it is synchronized with v_i, the energy generation and consumption of v_i at different synchronization time points, and the amount of data generated by v_i at each update round. Also, the digital twin DT_i of sensor v_i can emulate or predict the behaviour of sensor v_i such as its data generation since last data collection. To ensure that DT_i can simulate the behaviour of v_i accurately, the state of DT_i needs to be synchronized with v_i quite often, and any sensory data update of v_i must be sent to DT_i accordingly. Note that the DTs of different sensors may be stored at different cloudlets.

An illustrative example of the proposed network is given in Fig. 3.1, where the top figure is a wireless sensor network consisting of seven IoT devices, the bottom figure is the MEC network, and the DTs of IoT devices are placed in cloudlets that are colored with the green color.

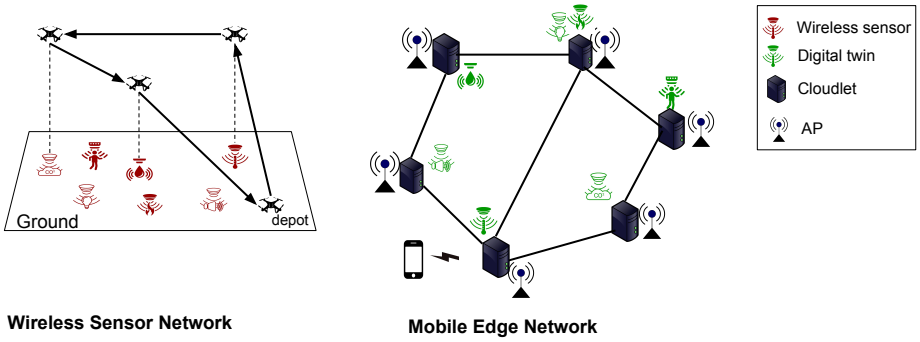


Figure 3.1: An illustrative example of an MEC network and DT placements.

3.2.2 Staleness of DT states

Let j_1 be the last synchronization time slot that DT_i was synchronized with sensor $v_i \in V$, and let j_2 be its next synchronization time slot with $j_2 > j_1$. The accumulative staleness $d_{v_i}(j_2)$ of the state of DT_i of sensor v_i from time slot j_1 to time slot j_2 is

$$d_{v_i}(j_2) = \sum_{t'=j_1}^{j_2} (t' - j_1 + 1) = \frac{(j_2 - j_1)(j_2 - j_1 + 1)}{2}. \tag{3.1}$$

Let $D(t)$ be the total staleness of DT states of all sensors at time slot t, then

$$D(t) = \sum_{v_i \in V} d_{v_i}(t), \tag{3.2}$$

where $d_{v_i}(t)$ is the accumulative number of time slots elapsed from the last synchronization of DT_i with sensor v_i to time slot t.

Let $l_{v_i}(t)$ be the number of time slots elapsed of DT_i of sensor v_i from its last

state synchronization to time slot t, Eq. (3.2) can be rewritten as follows,

$$D(t) = \sum_{v_i \in V} d_{v_i}(t) = \sum_{i=1}^{|V|} \frac{l_{v_i}(t) \cdot (l_{v_i}(t) + 1)}{2}. \tag{3.3}$$

Let $F(t)$ be the accumulative staleness of DT states of all sensors from time slot one to time slot t, then

$$F(t) = \sum_{t'=1}^{t} D(t') = \sum_{t'=1}^{t} \sum_{i=1}^{|V|} \frac{l_{v_i}(t') \cdot (l_{v_i}(t') + 1)}{2}. \tag{3.4}$$

It can be seen that function $F(t)$ is a strictly monotonic increasing function due to the fact that $F(t+1) - F(t) = D(t+1) \geq 0$.

For a given monitoring period T, to minimize the average staleness of DT states of all sensors in V at each time slot, the optimization objective is to

$$minimize \qquad \frac{F(T)}{T \cdot |V|}. \tag{3.5}$$

3.2.3 Energy model for a UAV

We assume that each UAV has energy capacity $\mathcal{E}_{UAV}$. It can upload its collected data and recharge itself at a depot s. The UAV can be used for data collection in a wireless sensor network, each tour of the UAV is a closed tour including depot s as it must return to the depot. The total energy consumption of each UAV tour should be no greater than its energy capacity $\mathcal{E}_{UAV}$. The UAV consumes energy on hovering for data collection at sensors and traveling above different sensors. For the sake of convenience, we assume that the UAV travels at a constant speed v, and let η_1 and η_2 be the costs of its energy consumptions on hovering and traveling per unit time, respectively.

3.2.4 Approximation ratio

Given a minimization optimization problem instance $\mathcal{P}$, let OPT and $S_{\mathcal{A}}$ be the values of the optimal solution and the approximation solution delivered by an approximation algorithm $\mathcal{A}$ for $\mathcal{P}$, respectively. The ratio ρ is referred to as the approximation ratio of the approximation algorithm if $\frac{S_{\mathcal{A}}}{OPT} \leq \rho$, i.e., the approximate solution is no greater than ρ times the optimal solution, and it can be seen that $\rho > 1$.

3.2.5 The staleness minimization problem of DT states

Definition 3. Given a renewable sensor network, each sensor $v_i \in V$ has a digital twin DT_i in a cloudlet of an edge computing network, a UAV with limited energy capacity is deployed for data collection of sensors and the collected data will be uploaded to the MEC, and the corresponding DTs of the sensors whose data have

been collected by the UAV at this round are updated accordingly. Given a finite time horizon T that is divided into equal time slots, *the staleness minimization problem of DT states* is to minimize the average staleness of DT states of all sensors that is defined in formula (3.5), subject to the energy capacity on the UAV, where the staleness of the DT state of a sensor is the number of time slots elapsed from its last update to the current time slot, and only some, not all DTs of sensors will be updated due to the fact that each UAV tour can only collect sensory data from some but not all sensors.

3.2.6 Fidelity-aware query evaluation problem on DTs

Built upon the DT data of sensors in the MEC network, we deal with user query services provided by the network service provider. Since different sensors have different DT states, the freshness of the collected data at the DTs are different too. Queries on the DT data at different time slots will result in different query results. For example, some query results may have high fidelity (or accurate), while others may have low fidelity due to the lack of real-time synchronizations between sensors and their DTs. We here deal with user queries on sensory data, by making use of the DT state information of sensors in cloudlets to develop efficient evaluation plans for fidelity-aware queries in the MEC network.

Specifically, assuming that there is a query for a subset of sensors $V' \subseteq V$ with specific requirements, such as the data staleness of most DTs of sensors in V' is no greater than δ time slots. For example, a user u issues a query $Q(t)$ at time slot t from an AP co-located with cloudlet n_u, which can be expressed by a tuple $Q(t) = (V', \delta, \vartheta)$, where the data collected from a sensor $v_i \in V'$ at DT_i is *fresh* if its last synchronization with sensor v_i is no greater than δ time slots; otherwise, the data stored at DT_i is not *fresh* by the requirement of query $Q(t)$. The set V' of sensors specified by query $Q(t)$ can then be partitioned into two disjoint subsets $V'_{>\delta} = \{v_i \mid v_i \in V', l_{v_i}(t) > \delta\}$ and $V'_{\leq\delta} = \{v_i \mid v_i \in V', l_{v_i}(t) \leq \delta\}$, where $l_{v_i}(t)$ is the staleness length of DT_i from its last synchronization with sensor v_i to time slot t. The value $|V'_{\leq\delta}|/|V'|$ is referred to as *the ratio of data freshness of the query result*, where *the fidelity* of a query result is defined as the ratio of the number of fresh sensors to all sensors in V'. An evaluation plan for query $Q(t)$ is *feasible* if the ratio of data freshness of the query result is no less than its pre-defined threshold ϑ, i.e.,

$$|V'_{\leq\delta}|/|V'| \geq \vartheta. \tag{3.6}$$

As the DTs of different sensors in V' are placed in different cloudlets, the collected data stored at their DTs will be aggregated for various query evaluations. Assume that the DT states (or data) of sensors placed in the same cloudlet will be aggregated prior to sending them to the *source cloudlet* $n_u \in \mathcal{N}$ of query $Q(t)$, i.e., we assume that each cloudlet contains only a single 'virtual' DT. If a cloudlet contains multiple DTs of sensors in V', it only sends the aggregation result of the DT data of the sensors to the source cloudlet once, where the aggregation result is referred to as the DT data of *the virtual DT* at that cloudlet, while the *source cloudlet* of a query is the

cloudlet (the colocated AP) from which the query is uploaded to the MEC network.

To find a cost effective evaluation plan for query $Q(t)$ issued by a user u under the coverage of AP (or source cloudlet) $n_u \in \mathcal{N}$ at time slot t, we aim to find an evaluation tree T_u in the MEC network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ rooted at cloudlet n_u and spanning all cloudlets that contain the DTs of sensors in V' such that the evaluation cost $C(Q(t))$ of query $Q(t)$ defined blow is minimized,

$$C(Q(t)) = \sum_{(x,y)\in E(T_u)} \rho(x,y) D^{(T_u)}(x,y) + \sum_{v \in V(T_u)} agg^{(T_u)}(v), \tag{3.7}$$

where $\rho(x,y)$ is the transmission cost per unit data along link (x,y) in the MEC network between cloudlets x and y, $D^{(T_u)}(x,y)$ is the volume of data to be transferred on link (x,y) in tree T_u, $agg^{(T_u)}(v)$ is the processing cost of aggregating data from the children of node v in tree T_u, and $V(T_u)$ is the set of nodes and $E(T_u)$ is the set of edges in T_u, respectively. All cloudlet nodes in T_u will participate in the query evaluation and consume their computing resources.

Assume that a cloudlet node v in T_u contains l children: $u_1, u_2, \ldots, u_l$. The processing cost $agg^{(T_u)}(v)$ at cloudlet v in tree T_u then is defined as follows.

$$agg^{(T_u)}(v) = \xi \cdot g(D^{(T_u)}(u_1, v), D^{(T_u)}(u_2, v), \ldots, D^{(T_u)}(u_l, v)), \tag{3.8}$$

where ξ is the processing cost per unit data at a cloudlet, while $D^{(T_u)}(u_{l'}, v)$ is the aggregate volume of data from the subtree $T_{u_{l'}}$ of T_u rooted at cloudlet node $u_{l'}$ with $1 \leq l' \leq l$. If node v is a leaf node, $agg^{(T_u)}(v) = \sum_{v_i \in V'} \{D_{v_i} \mid DT_i$ of sensor v_i is placed at clouldlet $v\}$, otherwise, $agg^{(T_u)}(v)$ is defined in Eq. (3.8) and $g(\cdot, \cdot, \ldots, \cdot)$ is an aggregation function that is determined by query $Q(t)$, and the volume of the aggregate result at a node usually is no more than the accumulative volume of data from its children in T_u.

Definition 4. Given the DTs of sensors in V deployed in cloudlets of an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a user u issues a fidelity-aware query $Q(t) = (V', \delta, \vartheta)$ at time slot t from an AP $n_u \in \mathcal{N}$ with $V' \subseteq V$. Recall that the query evaluation will be conducted in $\mathcal{G}$ and the query result will be sent back to user u through cloudlet $n_u \in \mathcal{N}$ (as the source cloudlet). *The fidelity-aware query evaluation problem* of $Q(t)$ in $\mathcal{G}$ is to (i) determine whether there is a feasible evaluation plan for $Q(t)$, and (ii) find a feasible evaluation plan for $Q(t)$ such that its evaluation cost $C(Q(t))$ defined in Eq. (3.7) is minimized if it does exist.

3.2.7 NP-hardness of the defined problems

Theorem 3.1. *The staleness minimization problem of DT states in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is NP-hard.*

Proof. We consider a special case of the problem when $T = 1$, in which case the problem is to synchronize as many sensors as possible with their DTs, by collecting their sensory data in one tour of the UAV. Since the energy capacity of the UAV is

limited, the UAV consumes energy on collecting data from each sensor in its tour and travelling between different sensors. The amount of energy consumed by the UAV at a sensor is determined by the volume of data generated by the sensor from its last collection (or the synchronization with its DT in the last round) to the current time slot. To this end, it is to find a closed tour for the UAV starting from and ending at depot s such that as many sensors as possible are included in the tour, subject to the energy capacity on the UAV, assuming that the UAV is in a depot s initially and will return to the depot after the tour. It can be seen that the travelling salesman problem (TSP) can be reduced to this special problem. While the TSP is a well known NP-hard problem, the staleness minimization problem is NP-hard too. ■ □

Theorem 3.2. *The fidelity-aware query evaluation problem in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is NP-hard.*

Proof. We consider a special case of the fidelity-aware query evaluation problem where a query $Q(t) = (V', \delta, \vartheta)$ has the data volume of the DT of every sensor in V' of 1, and the DTs of different sensors are deployed at different cloudlets. We then construct a multicast tree for the query rooted at the source cloudlet of the query such that its evaluation cost is minimized. This is equivalent to constructing a Steiner tree in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ spanning the source cloudlet and all cloudlet nodes containing the DTs of sensors in V', where the Steiner tree problem is to find the minimum-cost tree that connects a specific subset of the nodes in a graph. Since the Steiner tree problem is a well known NP-hard problem, the fidelity-aware query evaluation problem is NP-hard, too. ■ □

3.3 Optimal algorithm for a special staleness minimization problem of DT states

Since the synchronization budget (one UAV is used for data collection) at each time slot is given, this implies that not all DT states of sensors can be synchronized at each time slot. Then, it is challenging to identify which sensors to be included in the next UAV tour in order to minimize the average staleness of DT states of all sensors. In this section, we address the challenge by considering a special case of the staleness minimization problem, where there are the DTs of exactly K sensors can be synchronized with their sensors at each time slot t. That is, in a UAV tour at each time slot t, the UAV can choose K sensors among sensors for their data collection. Without loss of generality, let $v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}}$ be the sorted sensors in non-increasing order of their state staleness at time slot t with $l_{v_{i_1}}(t) \geq l_{v_{i_2}}(t) \geq \cdots \geq l_{v_{i_{|V|}}}(t)$. To minimize the objective function $\frac{F(T)}{T \cdot |V|}$ defined in formula (3.5), we choose to update the DT states of the first K sorted sensors per time slot t. We refer to this algorithm as `Algorithm` 5 as follows.

We claim that this will give an optimal solution to the special staleness minimization problem.

Algorithm 5 An optimal algorithm for the special staleness minimization problem of DT states.

Input: A set V of sensors with each sensor $v_i \in V$ having a DT_i in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a UAV with energy capacity located at depot s, and K DTs of sensors can be synchronized at time slot t with $1 \leq t \leq T$.

Output: A schedule of DT state synchronizations between sensors and their DTs at time slot t such that $F(t)$ is minimized with $1 \leq t \leq T$.

1: $S(t) \leftarrow \emptyset$; /* the solution at time slot t */
2: Calculate the staleness length $l_{v_i}(t)$ of DT_i of each sensor v_i in V;
3: Sort sensors in non-increasing order of the state staleness;
4: Add the first K sorted sensors to $S(t)$;
5: **return** $S(t)$ at time slot t.

Theorem 3.3. *Assuming that the DTs of sensors are stored in cloudlets in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, the update budget per update round is that the DT states of the first K sorted sensors are synchronized with their sensors at each time slot t with $1 \leq t \leq T$. Let $S(t)$ be the top-K sensors with the longest staleness length at time slot t delivered by the proposed algorithm, the solution $\sum_{t=1}^{T} S(t)$ delivered by the proposed algorithm for the special staleness minimization problem is optimal, where $K \leq |V|$.*

Proof. We first show that the solution $S(t)$ delivered by the proposed algorithm at each time slot t is optimal by induction, where $K \leq |V|$. It can be seen that at time slot $t = 1$, the claim is true. Assume that the claim is true for the first $t-1$ time slots, we now show that the claim is also true at time slot t by contradiction. Recall that the sorted sensor sequence is $v_1, v_2, \ldots, v_{|V|}$ in non-increasing order of their DT state staleness length. Assume that the DT state updates of sensors in the optimal solution $S'(t)$ at time slot t does not contain all the first K sorted sensors, let v_p be one of such a sensor with $p > K$, and let v_k be the first one among the K sorted sensors that is not in the optimal solution with $1 \leq k \leq K$. Following the definition of function l, we have $l_p(t) \leq l_k(t)$, another solution is obtained by replacing sensor v_p in the optimal solution with sensor v_k. The value difference between the new solution $S'(t) \cup \{v_k\} \setminus \{v_p\}$ and the optimal solution $S'(t)$ is $\frac{l_p(t) \cdot (l_p(t)+1)}{2} - \frac{l_k(t) \cdot (l_k(t)+1)}{2} \leq 0$, as $l_k(t) \geq l_p(t)$ by the assumption. Thus, we can replace each sorted sensor in the optimal solution whose index is not in $\{1, 2, \ldots, K\}$ by another sorted sensor whose index is in that range until all K sensors in the final solution are the first K sorted sensors. It can be seen that the final solution has the minimum value. This contradicts that solution $S'(t)$ is optimal.

Without loss of generality, let $S(t)$ be the solution delivered by `Algorithm 5` at time slot t. We then show that the solution $\cup_{t=1}^{T} S(t)$ is optimal by contradiction. Assume that the optimal solution to this special problem is $\cup_{t=1}^{T} S'(t)$, let $l(S'(t))$ be the sum of the DT staleness lengths of objects (sensors) in the optimal solution at time slot t. We have shown that $l(S(t)) \leq l(S'(t))$. Thus, $\sum_{t=1}^{T} l(S(t)) \leq \sum_{t=1}^{T} l(S'(t))$. We claim that $\sum_{t=1}^{T} l(S(t)) = \sum_{t=1}^{T} l(S'(t))$. Otherwise, this contradicts the assumption that $\cup_{t=1}^{T} S'(t)$ is the optimal solution. The theorem then follows. ■ □

3.4 Algorithm for the staleness minimization problem of DT states

In this section, we deal with the staleness minimization problem of DT states. We first develop an efficient approximation algorithm for a sub-problem of the problem. That is, given the update budget per update round, we find a closed tour for the UAV such that the optimization objective $F(t)$ defined in Eq. (3.4) is minimized, under an assumption that the volume of data generated by each sensor since its last data collection is given.

We approach this sub-problem through a reduction to the award collection maximization problem in an auxiliary graph. We then develop a deep learning mechanism to predict the volume of data generated by each sensor accurately by removing the assumption. We finally devise an efficient algorithm for the staleness minimization problem, which proceeds iteratively. Within each iteration t (at time slot t), we make use of the prediction result to find a closed tour in the auxiliary graph for the UAV to minimize the value of $F(t)$, and then adopt an adaptive strategy of data collection for the UAV to dynamically adjust the prediction error on the volume of data.

3.4.1 Approximation algorithm with the given volume of data generated by each sensor

We consider a subproblem to minimize $F(t)$, defined in Eq. (3.4), at time slot t through finding a close tour for the UAV such that as many sensors with large DT state staleness as possible can be included by the tour, assuming that the volume of data generated by each sensor is given. The general strategy for this subproblem is to reduce it to the award collection maximization problem in an auxiliary graph $G(t)$, and a closed tour in $G(t)$ subject to a given length is then found, which corresponds a feasible solution to the subproblem.

Before we proceed, we definite the orienteering problem and the award collection maximization problem as follows.

Given an undirected graph $G = (V, E; \pi, c)$ with $\pi : V \mapsto \mathbb{Z}^+$ and $c : E \mapsto \mathbb{Z}^+$, a source node $s \in V$, a destination $t \in V$, and a non-negative integer $L > 0$, *the orienteering problem* in G is to find a path $P_{s,t}$ from node s to node t such that the total reward $\sum_{v \in P_{s,t}} \pi(v)$ collected from the nodes in $P_{s,t}$ is maximized, while the length $\sum_{e \in P_{s,t}} c(e)$ of path $P_{s,t}$ is no greater than L, i.e., $\sum_{e \in P_{s,t}} c(e) \leq L$. The mentioned orienteering problem is NP-hard, and there is an approximation algorithm with an approximation ratio of 3 for it due to Bansal *et al.* [8], which is an improvement of the result in [12] if the edge weights in G abide by the triangle inequality.

Given the metric graph $G = (V, E; \pi, c)$, a non-negative value $\mathcal{E}_{UAV}$, and a specific node $s \in V$, *the award collection maximization problem* in G is to find a closed tour including node s such that the total award collected from the nodes in the closed tour is maximized, subject to that the tour length is upper bounded by $\mathcal{E}_{UAV}$.

Having the above preparations, we now construct an auxiliary graph $G(t) = (V \cup \{s\}, E(t); c(\cdot, \cdot), \pi(\cdot))$ for the optimization subproblem with the optimization

objective defined in Eq. (3.4) as follows.

The edge cost function $c : E(t) \mapsto \mathbb{R}^{\geq 0}$ and the node award function $\pi : V \cup \{s\} \mapsto \mathbb{Z}^{\geq 0}$ at each time slot t, node s is the depot of the UAV. There is an edge in $E(t)$ between any pair of nodes in $V \cup \{s\}$. For each edge $(u, v) \in E(t)$, its weight $c(u, v)$ is defined as follows.

$$c(u, v) = \eta_1 \cdot \frac{D_u(t) + D_v(t)}{2B} + \eta_2 \cdot \frac{dist(u, v)}{\nu}, \tag{3.9}$$

where $D_u(t)$ and $D_v(t)$ are the volumes of data generated by sensors u and v from their last data collection to time slot t, respectively, and $D_s(t) = 0$ of depot s. B is the radio bandwidth of the UAV, ν is the traveling speed of the UAV, $dist(u, v)$ is the Euclidean distance between two sensors u and v, and η_1 and η_2 are the amounts of energy consumed of the UAV on hovering and travelling per time unit. Note that cost $c(u, v)$ is the sum of energy consumption costs of the UAV at hovering on sensors u and v and travelling between them. Since edge (u, v) is contained in the closed tour, let (u', u) and (v', v) be the edges incident to nodes u and v in the closed tour respectively, then the energy consumption of the UAV for collecting data from sensor u is distributed to both edges (u', u) and (u, v) with each a half the amount of energy, and the energy consumption of the UAV for collecting data from sensor v is distributed to both edges (v', v) and (u, v) with each a half as well.

We aim to find a closed tour in $G(t)$ that includes depot s and as many sensors as possible, as each sensor in the tour will synchronize with its DT after the UAV finished the closed tour. Also, we expect the first K sorted sensors to be included in the closed tour to minimize the average staleness of DT states of all sensors in the next update round. Thus, a sensor with a longer DT state staleness should be assigned a larger award, i.e., each sensor node $v_i \in V$ in $G(t)$ is assigned the amount of award $\pi(v_i)$, which is its accumulative staleness of DT_i from its last update time slot to time slot t as follows.

$$\pi(v_i) = \frac{l_{v_i}(t) \cdot (l_{v_i}(t) + 1)}{2}. \tag{3.10}$$

It can be seen that $\pi(v_i) = d_{v_i}(t)$, where $d_{v_i}(t)$ is the accumulative number of time slots elapsed of sensor v_i from its last data collection time slot to time slot t that is defined in Eq. (3.1), and $l_{v_i}(t)$ is the number of time slots elapsed since its last data collection until time slot t. It can be shown that auxiliary graph $G(t)$ is a metric graph and its edge weights meet the triangle inequality.

The optimization subproblem with optimization objective $F(t)$, defined in Eq. (3.4), at time slot t now is reduced to *the award collection maximization problem* in $G(t)$, while the latter is to find a closed tour $\mathcal{C}(t)$ in $G(t)$ for the UAV including depot s so that the accumulative award collected by the UAV from sensors in the tour is maximized, subject to that the total energy consumption of the UAV in the tour is no greater than its energy capacity $\mathcal{E}_{UAV}$. It is well known that the award collection maximization problem is NP-hard, and there is a constant approximation algorithm for it due to Liang *et al.* [75].

Algorithm 6 An approximation algorithm for minimizing $F(t)$ with the given data volume assumption at time slot t.

Input: A set V of sensors with each sensor $v_i \in V$ having a DT_i in an MEC network $G = (\mathcal{N}, \mathcal{E})$, and a UAV with energy capacity $\mathcal{E}_{UAV}$ located at depot s, the volume $D_{v_i}(t)$ of data generated by each sensor $v_i \in V$ from its last data collection to time slot t is given with $1 \leq t \leq T$.

Output: A schedule of DT state synchronizations between sensors and their DTs at time slot t such that $F(t)$ is minimized with $1 \leq t \leq T$.

1: $S(t) \leftarrow \emptyset$; /* the solution at time slot t */
2: Calculate the staleness length of DT_i of each sensor v_i in V. Let $l_{v_i}(t)$ be the staleness length of DT_i from its last synchronization to time slot t;
3: Construct an auxiliary graph $G(t) = (V \cup \{s\}, E'(t); c(\cdot,\cdot), \pi(\cdot))$, where the weight $c(u,v)$ of edge (u,v) is the energy consumption of the UAV for hovering at sensors u and v and travelling between them that is defined in Eq. (3.9), and $\pi(v_i)$ $(= \frac{l_{v_i}(t)(l_{v_i}(t)+1)}{2})$ is the award of node $v_i \in V$ that is the accumulative staleness of DT_i from its last synchronization with v_i to time slot t;
4: Find a closed tour $\mathcal{C}(t)$ in graph $G(t)$ including depot s with maximizing the total award collected from sensors in the tour, by applying the approximation algorithm for the award collection maximization problem due to Liang *et al.* [75];
5: Collect data from sensors in $\mathcal{C}(t)$ by the UAV, and synchronize the DT state of each sensor in $\mathcal{C}(t)$ after the UAV uploaded all collected data at depot s;
6: $S(t) \leftarrow \mathcal{C}(t)$;
7: **return** The synchronization schedule $S(t)$ at time slot t.

The proposed approximation algorithm for the optimization subporblem - minimizing $F(t)$ defined in Eq. (3.4), is presented in `Algorithm` 6 under the assumption that the volume of data generated by each sensor is given.

3.4.2 Predicting the data volume of each sensor

In the construction of auxiliary graph $G(t)$, we assumed that the volume $D_{v_i}(t)$ of data generated by each sensor $v_i \in V$ is given. In fact, the volume $D_{v_i}(t)$ of sensor v_i from its last synchronization to time slot t is unknown. We here remove this assumption, and develop a deep learning algorithm to predict the volume of data generated by each sensor during this period. We then make use of the prediction result to determine the next UAV tour.

We adopt the Long Short-Term Memory method (LSTM) [46] on its DT_i to predict the volume of each sensor $v_i \in V$ through analyzing its historical traces of the data generated at different time slots. We observe that the volume of data generated by sensor v_i has strong temporal patterns at different time slots. Therefore, an efficient prediction method should leverage the temporal patterns of the data generated by each sensor. We make use of a deep neural network (DNN) for time series data predictions [46].

Let $\zeta_i(t)$ be the volume of data generated by sensor v_i at time slot t. We use the DNN of v_i to work as a non-linear function $f(\zeta_i(1), \zeta_i(2), \ldots, \zeta_i(t))$, and the output

of function $f(\cdot,\cdot,\ldots,\cdot)$ is the predicted volume $\hat{\zeta}_i(t+1)$ of data generated by sensor v_i at time slot $t+1$. To this end, the DNN consists of two layers: a LSTM unit layer that first summarizes the information of the time series into a hidden state vector h_t and a fully connected layer that maps the vector h_t into $\hat{\zeta}_i(t+1)$. The LSTM unit consists of a memory cell, an input gate, an output gate, and a forget gate and (3.11) to (3.16) describes the process of the LSTM unit in details [46].

$$for_t = \sigma(W_{for} \cdot [h_{t-1}, \zeta_i(t)] + b_{for}) \tag{3.11}$$
$$in_t = \sigma(W_{in} \cdot [h_{t-1}, \zeta_i(t)] + b_{in}) \tag{3.12}$$
$$\widetilde{\mathcal{C}}_t = tanh(W_{\mathcal{C}} \cdot [h_{t-1}, \zeta_i(t)] + b_{\mathcal{C}}) \tag{3.13}$$
$$\mathcal{C}_t = forget_t \odot \mathcal{C}_{t-1} + in_t \odot \widetilde{\mathcal{C}}_t \tag{3.14}$$
$$out_t = \sigma(W_{out} \cdot [h_{t-1}, \zeta_i(t)] + b_{out}) \tag{3.15}$$
$$h_t = out_t \odot tanh(\mathcal{C}_t), \tag{3.16}$$

where W_* and b_* are the learn-able weights of the neural network and here $*$ is a wildcard representing 'for', 'in', 'out', and '$\mathcal{C}$'. Functions $\sigma(\cdot)$ and $tanh(\cdot)$ are the sigmoid activation function and hyperbolic tangent activation function, respectively, $[\cdot,\cdot]$ concatenates two matrices together, and $\odot$ is the Hadamard product.

Eq. (3.11) defines the forget gate, which decides what information to be removed from the memory cell. Eq. (3.12) and Eq. (3.13) together define the input gate, which decides what information to be added into the memory cell. Eq. (3.14) updates the memory cell state by combining the values of the forget gate and the input gate. Eq. (3.15) defines the output gate. Eq. (3.16) combines the value of output gate and the information in memory cell to generate the hidden state vector h_t.

In the end, $\hat{\zeta}_i(t+1)$ is obtained by

$$\hat{\zeta}_i(t+1) = W_{FC} \cdot h_t + b_{FC}, \tag{3.17}$$

where W_{FC} and b_{FC} are the weights of the fully connected layer.

For each sensor $v_i \in V$, we first use its historical data volume at each time slot to train the learn-able parameters of its DNN. We then use the DNN to predict the volume of data generated by the sensor at each later time slot. We finally obtain the predicted volume $\hat{D}_{v_i}(t)$ of sensor v_i from its last synchronization to time slot t, by adding up the volumes of data generated at these time slots.

Let $\zeta_i(t)$ be the volume of data generated by sensor v_i at time slot t. Assume that we aim to predict $D_{v_i}(t_2)$ and let t_1 be the last time slot sensor v_i was synchronized. We feed $\zeta_i(1), \zeta_i(2), \ldots, \zeta_i(t_1)$ into the LSTM of v_i sequentially. The LSTM then outputs the predicted volume $\hat{\zeta}_i(t_1+1)$ of the data generated by sensor v_i at time slot t_1+1. We then feed $\hat{\zeta}_i(t_1+1)$ to the LSTM to obtain $\hat{\zeta}_i(t_1+2)$, and so on. This procedure continues until we obtain $\hat{\zeta}_i(t_2)$. $\hat{D}_{v_i}(t_2)$ then can be calculated, i.e., $\hat{D}_{v_i}(t_2) = \sum_{t'=t_1+1}^{t_2} \hat{\zeta}_i(t')$. The detailed algorithm is given in `Algorithm 7`.

Algorithm 7 Prediction algorithm of the volume of data generated by each sensor $v_i \in V$ at time slot t.

Input: Historical data traces of sensor $v_i \in V$ before timeslot t' and its corresponding trained LSTM stored at its DT_i in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.

Output: Predict the volume $\hat{D}_{v_i}(t)$ of data generated by sensor v_i with DT_i in the next synchronization, i.e., the data generated by sensor v_i from its last data collection to time slot t with $1 \leq t \leq T$.

1: Let $\zeta_i(1), \zeta_i(2), \ldots, \zeta_i(t')$ be the sequence of the volume of generated data of sensor v_i at each time slot before time slot t;
2: Feed $\zeta_i(1), \zeta_i(2), \ldots, \zeta_i(t')$ into the LSTM of sensor v_i sequentially, and the predicted volume $\hat{\zeta}_i(t'+1)$ of data generated by sensor v_i is then obtained;
3: $\hat{D}_{v_i}(t) \leftarrow \hat{\zeta}_i(t'+1)$;
4: **for** $k \leftarrow t'+1$ to $t-1$ **do**
5: Feed $\hat{\zeta}_i(k)$ into the LSTM to obtain $\hat{\zeta}_i(k+1)$;
6: $\hat{D}_{v_i}(t) \leftarrow \hat{D}_{v_i}(t) + \hat{\zeta}_i(k+1)$;
7: **return** $\hat{D}_{v_i}(t)$ for each sensor $v_i \in V$ at time slot t.

3.4.3 Algorithm for the staleness minimization problem of DT states

Having the prediction mechanism for the volume of data generated by each sensor, and the approximation algorithm for DT state synchronizations of some sensors with themselves by finding the next UAV tour, we now propose an efficient algorithm for the staleness minimization problem of DT states as follows.

As the volume of data of each sensor in the UAV tour is the predicted one, this predicted volume may be larger or smaller than the actual volume. If the actual data volume of a sensor is higher than its predicted one, the UAV only collects the predicted amount of data, leaving the rest of data uncollected; otherwise (the actual volume is less than the predicted one), the UAV stays above the sensor much longer than the duration for collecting all data from that sensor. This results in the wasting of the precise UAV energy. To reduce unnecessary energy consumption and maximize data collection of the UAV, we adopt the following adaptive strategy of data collection for the UAV in tour $\mathcal{C}(t)$.

Let $v_0, v_1, v_2, \ldots, v_{m-1}$ be the sensor listed in their visiting order in tour $\mathcal{C}(t)$ with $v_0 = s$. The *planed energy consumption budget* $\hat{\mathcal{E}}(v_i, t)$ of the UAV when visiting sensor v_i in $\mathcal{C}(t)$ is defined as

$$\hat{\mathcal{E}}(v_i, t) = \sum_{j=0}^{i} (\eta_1 \cdot \hat{D}_{v_j}(t)/B + \eta_2 \cdot \frac{dist(v_j, v_{(j+1) \mod m})}{sp}), 0 \leq i \leq m-1 \text{ and } m \leq |V| \tag{3.18}$$

where $\hat{D}_{v_i}(t)$ is the predicted volume of data generated by sensor $v_i \in V$ from its last data collection to time slot t.

Assuming that the UAV is currently visiting sensor v_i for its data collection. If the UAV finishes data collection from sensor v_i before its allocated time duration at v_i, it moves to the next sensor $v_{(i+1) \mod m}$ for data collection; otherwise (the pre-

Algorithm 8 An algorithm for the staleness minimization problem of DT states.

Input: A set V of sensors with each sensor $v_i \in V$ having a DT_i in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, and a UAV with energy capacity $\mathcal{E}_{UAV}$ located at depot s.

Output: A schedule of DT state synchronizations between sensors and their DTs at each time slot t with $1 \leq t \leq T$.

Assume that $D_{v_i}(1)$ of each sensor $v_i \in V$ is given;
$t \leftarrow 1$; $S \leftarrow \emptyset$;
while $t \neq T + 1$ **do**
    Find a closed tour $\mathcal{C}(t)$ in $G(t)$ including depot s for the UAV with maximizing the total award, by applying the approximation algorithm, Algorithm 6, and let $v_0, v_1, \ldots, v_{m-1}$ be the sequence of sensors in $\mathcal{C}(t)$ with $v_0 = s$;
    Calculate the planned energy consumption budget $\hat{\mathcal{E}}(v_i)$ of each sensor $v_i \in \mathcal{C}(t)$;
    for each $v_i \in \mathcal{C}(t)$ **do**
        Collect data from sensor v_i in tour $\mathcal{C}(t)$ by the UAV;
        if no more data can be collected from v_i within its allocated time duration **then**
            The UAV moves to the next sensor for data collection;
        else
            The actual volume of data of v_i is larger than the predicted one, the UAV continues collecting data from v_i until reaching its planned energy consumption budget $\hat{\mathcal{E}}(v_i, t)$ or no more data can be collected from v_i;
        Reset the data collection duration at sensor $v_i \in \mathcal{C}(t)$ by the above adjustment;
    Let $\mathcal{C}'(t)$ be the updated closed tour for the UAV with the adjusted hovering time at each sensor in the tour;
    The UAV uploads its collected data to the MEC network $\mathcal{G}$ at depot s, and the DT states of sensors in $\mathcal{C}'(t)$ are synchronized with their sensors;
    $S \leftarrow S \cup \{\mathcal{C}'(t)\}$;
    Predict the volume $\hat{D}_{v_i}(t+1)$ of data generated by each sensor $v_i \in V$ on the LSTM of v_i from its last data collection to time slot $t+1$, by applying `Algorithm 7`;
    $t \leftarrow t + 1$;
return The synchronization schedule S from time slot 1 to time slot T.

dicted volume of data is no larger than the actual volume of data of sensor v_i), if the actual energy consumption $\mathcal{E}(v_i, t)$ of the UAV so far is less than its planned energy consumption budget $\hat{\mathcal{E}}(v_i, t)$, it continues data collection from sensor v_i until its planned energy consumption budget $\hat{\mathcal{E}}(v_i, t)$ reaches or no more data can be collected from sensor v_i. The total energy consumption of the UAV is upper bounded by $\mathcal{E}_{UAV}$ when it finishes the tour, and the accumulative volume of uncollected data from sensors in the tour can be significantly reduced, which is demonstrated in the later empirical analysis.

The detailed algorithm for the staleness minimization problem of DT states is given in `Algorithm 8`.

3.4.4 Algorithm analysis

Theorem 3.4. *There is a* $(1 + \frac{2\beta}{3})$*-approximation algorithm* `Algorithm` *6, for the special optimization subproblem with optimization objective* $F(t)$ *defined in* (3.4)*, under the assumption that the volume of data generated by each sensor since its last data collection is given, and the solution delivered by* `Algorithm` *6 is feasible, where* $OPT(t)$ *is the optimal value of function* $F(t)$*,* $A^*(t)$ *is the maximum value of the accumulative award collected by a UAV along an optimal closed tour, and* $\beta = \max\{A^*(t)/OPT(t) \mid 1 \leq t \leq T\}$ *with* $\beta > 1$*.*

Theorem 3.5. *There is an efficient algorithm* `Algorithm` *8 for the DT staleness minimization problem of DT states in an MEC network* $\mathcal{G} = (\mathcal{N}, \mathcal{E})$*, and the solution delivered by the algorithm is feasible.*

Proof. The feasibility of the solution obtained by `Algorithm` 8 can be shown exactly as we did for Theorem 3.4, i.e., the total energy consumption of the UAV per tour is upper bounded by $\mathcal{E}_{UAV}$ if we adopt the adaptive strategy of data collection for predicting errors correction.

Since the UAV tour at time slot t is based on the prediction of the volume of data generated by each sensor from its last data collection to time slot t, such a prediction may not be 100% accurate and the prediction error is unavoidable, this implies that the actual volumes of data at some sensors may be higher or lower than their predicted ones. In the proposed algorithm `Algorithm` 8, we adopt an adaptive strategy of data collection for the UAV to mitigate this prediction error. We have shown that the total energy consumption of the UAV in the closed tour is upper bounded by its energy capacity $\mathcal{E}_{UAV}$. Thus, the solution delivered by `Algorithm` 8 is feasible, and the accumulative volume of uncollected data is significantly reduced in comparison to the case without adopting the adaptive strategy of data collection in later empirical evaluation. The solution thus is feasible. □

3.5 Fidelity-aware query evaluation on DT data

We consider fidelity-aware query evaluation, built upon the DT data of sensors in cloudlets in $\mathcal{G}$. We first show that the freshness of DT states of sensors impacts on the fidelity (data freshness) of the query result. We then develop a cost-effective evaluation plan for fidelity-aware queries.

Recall that query $Q(t) = (V', \delta, \vartheta)$ is issued by user u from AP n_u at time slot t. Assume that the query result of $Q(t)$ will be returned to cloudlet n_u as well. As the DTs of sensors in V' are placed in different cloudlets, the collected data from these sensors stored at their DTs will be aggregated for query evaluation. Assume that the DT data of sensors in the same cloudlet will be sent to the source cloudlet n_u by their aggregate result along a routing path, i.e., we assume that each cloudlet contains at most one single virtual DT. A cloudlet containing multiple DTs of sensors in V' will be treated as a single virtual DT.

3.5.1 Evaluation trees for fidelity-aware queries

To minimize the evaluation cost of query $Q(t)$, we will find an evaluation tree rooted at cloudlet n_u spanning cloudlets that contain the DTs of sensors in V' such that the evaluation cost of query $Q(t)$ is minimized.

In the following, we detail the construction of tree T_u. Let $\mathcal{N}'$ be the set of cloudlets that contain the DTs of sensors in V' of $Q(t)$. We further assume that the cloudlets in $\mathcal{N}'$ are sorted in non-increasing order of the accumulative volume of DT data of sensors in V' in them, i.e., the cloudlet sequence is $n_{c_1}, n_{c_2}, \ldots, n_{c_{|\mathcal{N}'|}}$, where cloudlet n_{c_1} contains the largest accumulative volume of DT data of sensors in V', while cloudlet $n_{c_{|\mathcal{N}'|}}$ contains the least accumulative volume of DT data of sensors in V', respectively.

The tree T_u is constructed greedily. At each time, a cloudlet node in $\mathcal{N}'$ with the minimum evaluation cost gain will be added to T_u. This procedure continues until all cloudlet nodes in $\mathcal{N}'$ are added to tree T_u. The construction of T_u is obtained through constructing a series of partial evaluation trees $T(u,k)$ that spans k cloudlets nodes with root at cloudlet $n_u \in \mathcal{N}$, where $T(u,k)$ is a partial evaluation tree of T_u spanning k cloudlet nodes in $\mathcal{N}'$.

When $k = 1$, the partial evaluation tree $T(u,1)$ is a shortest path P_1 in the MEC network $\mathcal{G}(\mathcal{N},\mathcal{E})$ from cloudlet n_{c_1} to cloudlet n_u, where each edge $(v_i, v_{i+1}) \in P_1$ with $v_1 = n_{c_1}$, $v_{|P_1|+1} = n_u$, and v_{i+1} is the parent of v_i in the partial evaluation tree. The accumulative volume of data on edge (v_i, v_{i+1}) in P_1 is $agg^{(T(u,1))}(v_i) = D_{c_1}$, which is the accumulated volume of DT data of sensors in V' stored at cloudlet n_{c_1}, where $agg^{(T(u,1))}(v)$ is an aggregate function related to query $Q(t)$ at node v in tree $T(u,1)$. The communication cost on tree edge (v_i, v_{i+1}) is $\rho(v_i, v_{i+1}) \cdot agg^{(T(u,1))}(v_i)$, where $\rho(v_i, v_{i+1})$ is the communication cost of one unit data transfer on tree edge (v_i, v_{i+1}).

Assuming that tree $T(u,k-1)$ has been constructed with $k > 1$, we now construct $T(u,k)$ by adding a cloudlet node $n_{c_i} \in \mathcal{N}'$ to the tree with the minimum cost gain $\Delta(T(u,k-1), n_{c_i}, n_{c_j})$, where n_{c_j} is the tree node connecting cloudlet node $n_{c_i} \in \mathcal{N}'$ through a shortest path $P_k(n_{c_i}, n_{c_j})$ in $G \setminus T(u,k-1)$ that has not been contained in $T(u,k-1)$ yet. Let $P_k^T(n_{c_j}, n_u)$ be the tree path in $T(u,k-1)$ between node n_{c_j} and node n_u. For the volume of data to be routed along a routing path from n_{c_i} to n_u, the routing path can be further divided into two segments $P_k(n_{c_i}, n_{c_j})$ from n_{c_i} to n_{c_j} and $P_k^T(n_{c_j}, n_u)$ in tree $T(u,k-1)$ from n_{c_j} to n_u, respectively. The data volume on each link in $P_k(n_{c_i}, n_{c_j})$ is D_{c_i}, while the data volume on each link in $P_k^T(n_{c_j}, n_u)$ is the aggregation result of the volume D_{c_i} with other existing ones in $T(u,k-1)$. The evaluation cost gain $\Delta(T(u,k-1), n_{c_i}, n_{c_j})$ of $T(u,k)$ built upon tree $T(u,k-1)$, by

adding a shortest path from cloudlet node n_{c_i} to a tree node n_{c_j} thus is defined as

$$\begin{aligned}\Delta(T(u,k-1),n_{c_i},n_{c_j}) = & \sum_{(x,y)\in P_k^T(n_{c_i},n_u)}^{y=p(x)} (\rho(x,y)D^{(T(u,k))}(x,y) + agg^{(T(u,k))}(y)) \\ & - \sum_{(x,y)\in P_{k-1}^T(n_{c_j},n_u)}^{y=p(x)} (\rho(x,y)D^{(T(u,k-1))}(x,y) + agg^{(T(u,k-1))}(y)) \\ & = \sum_{(x,y)\in P_k(n_{c_i},n_{c_j})}^{y=p(x)} (\rho(x,y)D^{(T(u,k))}(x,y) + agg^{(T(u,k))}(y)) \\ & + \sum_{(x,y)\in P_k^T(n_{c_j},n_u)}^{y=p(x)} (\rho(x,y)D^{(T(u,k))}(x,y) + agg^{(T(u,k))}(y)) \\ & - \sum_{(x,y)\in P_{k-1}^T(n_{c_j},n_u)}^{y=p(x)} (\rho(x,y)D^{(T(u,k-1))}(x,y) + agg^{(T(u,k-1))}(y)), \end{aligned} \tag{3.19}$$

where $P_{k-1}^T(n_{c_j},n_u)$ is the tree path in $T(u,k-1)$, $y=p(x)$ implies that node y is the parent of node x in the tree. Function $agg^{(T(u,k))}(y)$ at node $y \in P_k^T(n_{c_j},n_u)$ in tree $T(u,k)$ now adds one more data volume D_{c_i} from cloudlet n_{c_i} for aggregation in addition to the original ones from its children in $T(u,k-1)$. The value of $D^{(T(u,k))}(x,y)$ of edge (x,y) in $T(u,k)$ is the aggregate volume of data transferred from x to y, which is no less than its value in tree $T(u,k-1)$ and determined by the aggregate function $g(\cdot)$.

The construction of the evaluation tree T_u for query $Q(t)$ proceeds as follows. It contains only a cloudlet node n_u as its root initially. Cloudlet node n_{c_1} is then added to T_u, i.e., tree $T(u,1)$ is built. Each other cloudlet node $n_{c_i} \in \mathcal{N}' \setminus \{n_{c_1}\}$ is then added to T_u iteratively. In iteration k with $2 \le k \le |\mathcal{N}'|$, a cloudlet node $n_{c_{i_0}} \in \mathcal{N}'$ with the minimum evaluation cost gain in Eq. (3.19) is added to the current partial evaluation tree $T(u,k-1)$ until all cloudlet nodes in $\mathcal{N}'$ are added. The evaluation tree $T_u = T(u,|\mathcal{N}'|)$ is then constructed with $|\mathcal{N}'| \le |\mathcal{N}|$, and the detailed algorithm for the construction of T_u is presented in `Algorithm` 9.

3.5.2 Algorithm for fidelity-aware query evaluation

Having the query evaluation tree T_u for query $Q(t)$ of user u issued at time slot t, the rest is to examine whether the evaluation plan T_u is feasible, by collecting the DT information of sensors in V' and forwarding the information to cloudlet n_u along tree paths of T_u. The proposed algorithm for the fidelity-aware query evaluation problem is given in `Algorithm` 10.

Algorithm 9 The construction of an evaluation tree T_u for query $Q(t)$ rooted at cloudlet $n_u \in \mathcal{N}$ in MEC $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.

Input: An MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{C})$ and a query $Q(t) = (V', \delta, \vartheta)$, assuming cloudlets that contain the DTs of sensors in V' are sorted as $n_{c_i}, n_{c_2}, \ldots, n_{c_{|\mathcal{N}'|}}$ in non-increasing order of the data volume of DTs, where cloudlet n_{c_i} has a data volume D_{c_i}, then $D_{c_1} \geq D_{c_2} \geq \ldots \geq D_{c_{|\mathcal{N}'|}}$, $1 \leq c_i \leq |\mathcal{N}|$ and $1 \leq i \leq |\mathcal{N}|$.

Output: An evaluation tree T_u rooted at n_u and spanning all cloudlets that contain the DTs of sensors in V' such that the evaluation cost $C(Q(t))$ of query $Q(t)$ for user u is minimized.

$C(Q(t)) \leftarrow \infty$; /* the solution cost*/
$T_u \leftarrow \emptyset$; /* the query evaluation tree */
$\mathcal{N}' \leftarrow \{n_{c_1}, n_{c_2}, \ldots, n_{c_{|\mathcal{N}'|}}\}$;
Find a shortest path P_1 in the MEC network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ from n_{c_1} to n_u;
$T_u \leftarrow P_1$; $k \leftarrow 1$; $\mathcal{N}' \leftarrow \mathcal{N}' \setminus \{n_u\}$; $T(u, 1) \leftarrow P_1$;
while $\mathcal{N}' \neq \emptyset$ **do**
    $k \leftarrow k + 1$;
    Construct the partial evaluation tree $T(u, k)$ from tree $T(u, k-1)$ as follows;
    for each $n_{c_i} \in \mathcal{N}'$ **do**
        for each $n_{c_j} \in T_u$ **do**
            Find a shortest path $P_k(n_{c_i}, n_{c_j})$ in $\mathcal{G} \setminus T(u, k-1)$ between cloudlets n_{c_i} and n_{c_j};
            Compute the costs of segments $P_k(n_{c_i}, n_{c_j})$ and $P_k^T(n_{c_j}, n_u)$ of $P_k^T(n_{c_i}, n_u)$;
            Construct a potential partial evaluation tree $T(u, k) \leftarrow T(u, k-1) \cup P_k(n_{c_i}, n_{c_j})$, by connecting cloudlet node n_{c_i} to tree $T(u, k-1)$ through the tree node n_{c_j} with a shortest path $P_k(n_{c_i}, n_{c_j})$;
            Calculate $\Delta(T(u, k-1), n_{c_i}, n_{c_j})$, the evaluation cost gain by Eq. (3.19);
    $i_0, j_0 \leftarrow \min \arg_{n_{c_i} \in \mathcal{N}',\ n_{c_j} \in T(u,k-1)} \{\Delta(T(u, k-1), n_{c_i}, n_{c_j})\}$; /* choose cloudlet node $n_{c_{i_0}}$ with the minimum cost gain and add it to $T(u, k-1)$ */;
    $T(u, k) \leftarrow T(u, k-1) \cup P_k(n_{c_{i_0}}, n_{cj_0})$; /* adding cloudlet node $n_{c_{i_0}}$ to the multicast tree */
    $\mathcal{N}' \leftarrow \mathcal{N}' \setminus \{n_{c_{i_0}}\}$;
$T_u \leftarrow T(u, k)$;
$C(Q(t)) \leftarrow C(T_u)$;
return Solution T_u with evaluation cost $C(T_u)$.

3.5.3 Algorithm analysis

Theorem 3.6. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, and the DTs of sensors in a sensor network are stored in its cloudlets, users can issue a fidelity-aware query $Q(t) = (V', \delta, \vartheta)$ on the DT data. There is an efficient algorithm,* `Algorithm` *10, for the fidelity-aware query evaluation problem in $\mathcal{G}$, which delivers a feasible evaluation plan if such a plan does exist. The algorithm takes $O(|\mathcal{E}|^2 \cdot |\mathcal{N}|^2 \cdot \log\ |\mathcal{N}|)$ time.*

Proof. It can be seen that if the evaluation plan is feasible, the solution delivered by

Algorithm 10 A query evaluation algorithm.

Input: An MEC network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ and a query $Q(t) = (V', \delta, \vartheta)$ issued by user u under the coverage of AP n_u at time slot t with $1 \leq t \leq T$, assuming that cloudlets containing the DTs of sensors in V' are sorted as a list $n_{c_i}, n_{c_2}, \ldots, n_{c_{|\mathcal{N}'|}}$ in non-increasing order of the accumulative volume of DT data in them, i.e., each cloudlet n_{c_i} has a data volume D_{c_i} and $D_{c_1} \geq D_{c_2} \geq \ldots \geq D_{c_{|\mathcal{N}'|}}$, where $1 \leq c_i \leq |\mathcal{N}'|$ and $1 \leq i \leq |\mathcal{N}|$.

Output: An evaluation tree T_u for $Q(t)$ rooted at n_u and spanning cloudlets in $\mathcal{N}'$ such that the evaluation cost $C(Q(t))$ of $Q(t)$ is minimized if T_u is feasible.

1: Find a cost-effective evaluation plan – an evaluation tree T_u for $Q(t)$ by invoking `Algorithm` 9;
2: Aggregate DT data by the aggregate function $agg(\cdot)$ of $Q(t)$ at each cloudlet node in T_u from leaves to the tree root n_u;
3: Identify the subset $V'_{\leq \delta}$ from set V' at cloudlet n_u;
4: **if** $|V'_{\leq \delta}|/|V'| \geq \vartheta$ **then**
5: A cost-effective feasible evaluation plan T_u for query $Q(t)$ is obtained
6: **else**
7: No solution can meet the query requirement; EXIT.

`Algorithm` 10 is feasible. The rest is to analyze the time complexity of the evaluation algorithm. The dominant time complexity in `Algorithm` 10 is the construction of the evaluation tree T_u, which consists of $|\mathcal{N}'|$ iterations (with $|\mathcal{N}'| \leq \mathcal{N}|$ and $\mathcal{N}'$ is the set of cloudlets containing the DTs of sensors in V'). Within each iteration, it finds a shortest path in $\mathcal{G}$ that takes $O(|\mathcal{N}| \cdot |\mathcal{E}| \cdot \log\ |\mathcal{N}|)$ time. Thus, the the time for the construction of tree T_u is $O(|\mathcal{E}| \cdot |\mathcal{N}|^2 \cdot \log\ |\mathcal{N}|)$. The determination whether tree T_u is a feasible evaluation plan for fidelity-aware query $Q(t) = (V', \delta, \vartheta)$ can be done within $O(|\mathcal{N}|)$ time, by collecting the DT data of sensors in V' and aggregating the data at tree nodes. The query evaluation algorithm takes $O(|\mathcal{E}| \cdot |\mathcal{N}|^2 \cdot \log\ |\mathcal{N}|)$ time. □

3.6 Performance evaluation

In this section, we evaluated the proposed algorithms through experimental simulations, using a real data set. We also investigated the impact of important parameters on the performance of the proposed algorithms.

3.6.1 Experimental settings

We consider an MEC network that consists of 20 APs with each having a co-located a cloudlet. The MEC network is generated by GT-ITM [63]. We assume that the transmission cost on a link between two APs is randomly drawn from \$0.1 to \$0.4 per MB [64]. Also, the processing cost of cloudlets is randomly drawn from \$0.04 to \$0.06 per MB [63]. There are 500 sensors randomly deployed in a circular area with a 500 meter radius and a depot s located at the center of the area. The digital twins of sensors are randomly assigned to cloudlets. A UAV is deployed at depot s initially.

For the special staleness minimization problem of DT states, we assume that $K = 100$ sensors per UAV tour. For the staleness minimization problem of DT states, the UAV has energy capacity $\mathcal{E} = 3 \times 10^5$ *joules* at constant flying speed $v = 10$ m/s. The energy consumption rates of the UAV on hovering and traveling are $\eta_1 = 150$ J/s and $\eta_2 = 100$ J/s, respectively [71]. The data transmission rate is 1 *Mbps* [157]. The monitoring period consists of 1,000 time slots. Moreover, we use a real-world dataset: California traffic usage dataset collected from 17,544 sensors on the San Francisco Bay area freeways [56]. Each data point in the dataset is a time series of road occupy rates (between 0 and 1) hourly recorded by a sensor. To evaluate the performance of the proposed algorithms, we use the road occupy rate data to simulate the volume of data generated by each sensor. We assume that a sensor can generate at most $[5, 10]$ MB data per time slot [157]. Then, for each sensor, we select a time series in the California traffic usage dataset and map the road occupy rate to the generated data volume by multiplying it by the maximum volume of generated data.

To evaluate the performance of the optimal algorithm, `Algorithm` 5, for the special case of the problem by choosing top-K sensors, we propose a baseline `Random`, where K sensors are randomly picked to synchronize at each time slot. We evaluate `Algorithm` 8 against a heuristic `Heur` that proceeds as follows. At each time slot, we construct the closed tour of the UAV greedily, and the tour contains only depot s initially. The sorted sensors then are added to the tour one by one if the addition does not violate the energy capacity of the UAV; otherwise, the next sensor in the sorted sequence is considered. This procedure continues until all sensors are considered.

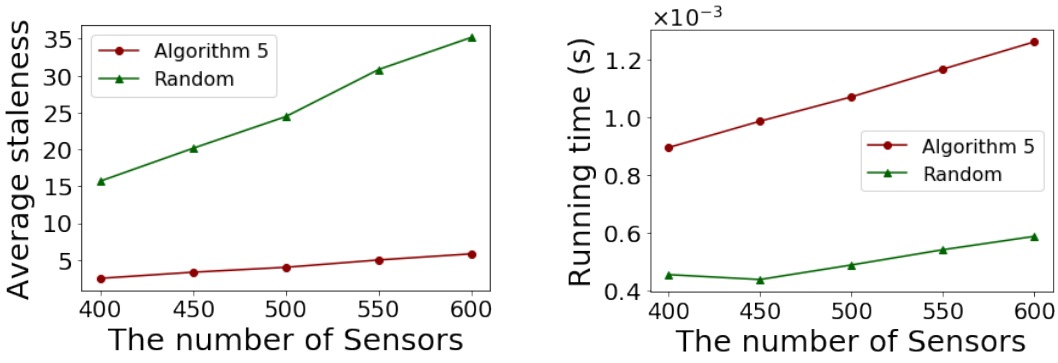
To evaluate the adaptive strategy of data collection for the staleness minimization problem of DT states, we consider a variant of `Algorithm` 8 without adopting the policy, referred to as `NoPolicy`, to examine how much data left without collection per tour. To study the performance of the proposed algorithm for the problem, we refer to `Algorithm` 6 and `Algorithm` 8 as `Actual` and `Pred` with and without the prediction of the volume of data generated by each sensor since its last data collection.

For the fidelity-aware query evaluation problem, we assume that each query requests data from the digital twins of 4-8 sensors [63]. We set the threshold ratio ϑ at 0.8. The threshold δ is set at 1. The aggregation function is randomly chosen from a family of aggregation functions, e.g., `{sum, average, max, min}` [63].

The value in each figure is the mean of the results out of 50 network instances of the same size. The running time of an algorithm is obtained based on a machine with 3.6 GHz Intel i7 single-core CPU and 16 GB RAM. Unless otherwise specified, the parameters are adopted in the default setting.

3.6.2 Performance evaluation of the staleness minimization algorithm

We started with investigating the performance of different algorithms for the special staleness minimization problem of DT states by varying the number of sensors from 400 to 600. In Fig. 3.2 (a), we can see that `Algorithm` 5 delivers a solution with a lower average staleness than that of algorithm `Random`, and Fig. 3.2 (b) illus-

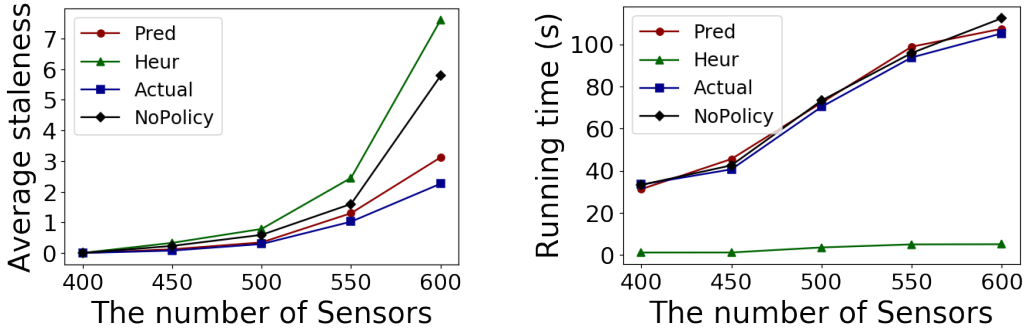


(a) Average staleness (b) Average running time

Figure 3.2: Performance of different algorithms for the special staleness minimization problem of DT states by varying the number of sensors.

trates their running time.

We then studied the performance of different algorithms for the staleness minimization problem of DT states, by varying the number of sensors from 400 to 600. Fig. 3.3 (a) illustrates the average staleness delivered by different algorithms. It can be seen that the staleness of DT states delivered by `Pred` is much smaller than that of algorithm `Heur`. With the increase on network size, the staleness gap between `Pred` and algorithm `Heur` enlarges, which indicates that `Pred` utilizes the energy of the UAV efficiently. In addition, even if we assume that the volume of data generated by each sensor is given, `Actual` only decreases the average staleness of all sensors by 15.30%, compared with `Pred` when there are 500 sensors. This demonstrates the effectiveness of the proposed prediction algorithm.



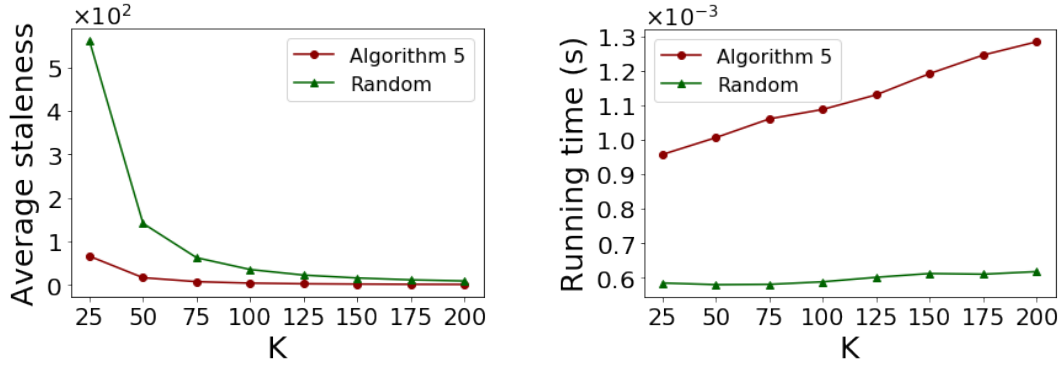
(a) Average staleness (b) Average running time

Figure 3.3: Performance of different algorithms for the staleness minimization problem of DT states varying the number $|V|$ of sensors.

Moreover, algorithm `NoPolicy` increases the average DT staleness by 70.84% compared with `Pred`. The rationale behind is that the UAV wastes more energy on hovering above sensors even after the data collection from the sensors has been finished under the schedule of algorithm `NoPolicy`.

3.6.3 The impact of important parameters on the performance of different algorithms

We first investigated parameter K on the performance of different algorithms for the special staleness minimization problem of DT states by varying the value of K from 25 to 200. It can be seen from Fig. 3.4(a) that with the increase of K, the average staleness of DTs in the solutions delivered by both algorithms decreases, since more DTs can be synchronized with their corresponding sensors. With the increase of K, the average staleness gap between the two algorithms becomes smaller. Fig. 3.4(b) plots the running times of these two algorithms.

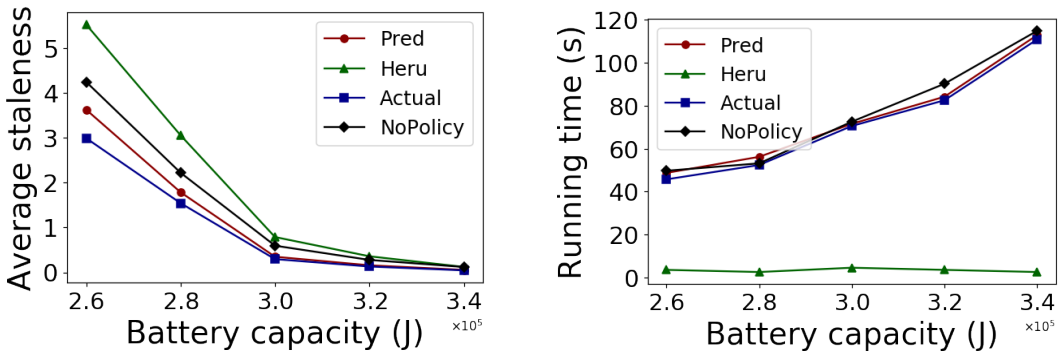


(a) Average staleness (b) Average running time

Figure 3.4: Performance of different algorithms for the special staleness minimization problem of DT states varying K.

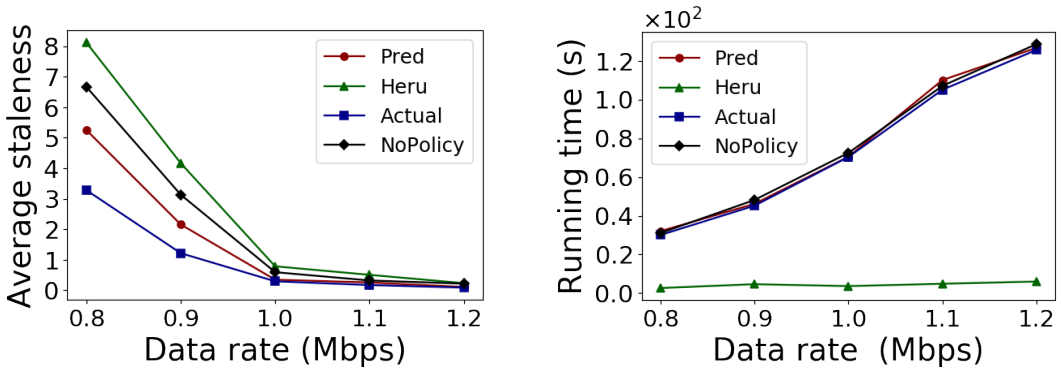
We then evaluated the impacts of parameters $\mathcal{E}_{UAV}$ and B on the performance of the proposed algorithms for the staleness minimization problem of DT states, by varying $\mathcal{E}_{UAV}$ from 2×10^5 to 4×10^5 *Joules*. Fig. 3.5 (a) presents the impact of the energy capacity on the average DT staleness. The staleness reduces with the growth on the energy capacity, as the UAV can visit more sensors and these sensors can synchronize with their DTs. It can be seen that `Pred` is superior to algorithms `Heur` and `NoPolicy` respectively. The staleness difference becomes smaller when the UAV has more energy capacity. This can be justified that the UAV is more tolerant to inaccurate predictions of the data volume at sensors when it has a large energy capacity. Fig. 3.5 (b) plots the running time of the comparison algorithms.

The rest is to study the impact of the data rate B of the UAV by varying it from 0.8 *Mbps* to 1.2 *Mbps*. Fig. 3.6 (a) shows that the average staleness DTs in the solutions delivered by all algorithms decreases with the increase on the data rate of the UAV. This is due to the fact that the growth on the data rate of the UAV leads to less hovering time above sensors, and thus more sensors can be visited by the UAV per tour. When the data rate is 0.8 *Mbps*, the average staleness of DTs by algorithm `Heur` is 8.12, which is approximately 54.67% more than that by `Pred`. Fig. 3.6 (b) describes the running times of different algorithms. Although `Pred` takes a longer running time than that of algorithm `Heur`, the average staleness in the solution delivered by `Pred` is much smaller than that by `Heur`.



(a) Average staleness (b) Average running time

Figure 3.5: Performance of different algorithms for the staleness minimization problem of DT states by varying the energy capacity $\mathcal{E}_{UAV}$ of sensors.



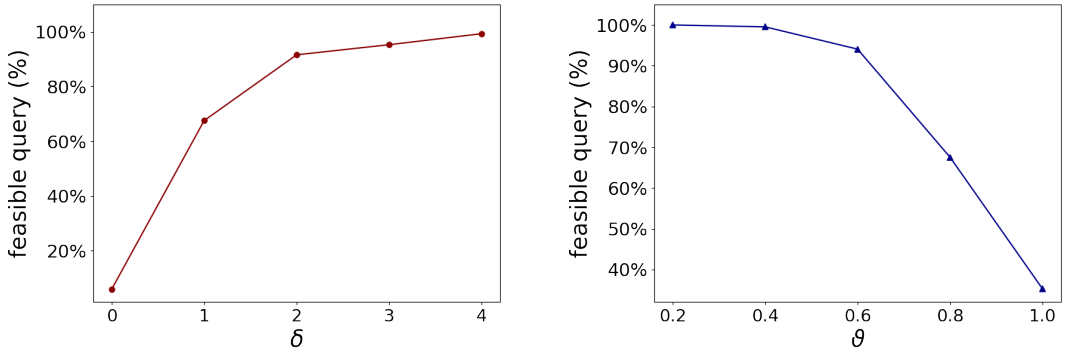
(a) Average staleness (b) Average running time

Figure 3.6: Performance of different algorithms for the staleness minimization problem of DT states by varying the data rate B of the UAV.

3.6.4 Performance evaluation of fidelity-aware query evaluation

To study the efficiency of `Algorithm` 10, we construct another evaluation tree, which is a minimum spanning tree rooted at source cloudlet $n_u \in \mathcal{N}$ initially, we then remove its leaves that do not contain the DT states of sensors in V' until all leaves in the resulting tree are the cloudlets containing the DT states of sensors in V'. Denote by `MST` the resulting tree as the query evaluation tree of user u.

We first evaluated a fidelity-aware query by setting the value of δ as 0, 1, 2, 3, and 4, respectively, and for each value of δ, we randomly generate 1,000 queries and examine the number of feasible evaluation plans among the generated queries. As shown in Fig. 3.7(a), the percentage of feasible evaluation plans for queries increases, with the growth of the value of δ. The rationale behind is that more DTs can meet the fidelity requirements of queries when δ is large. When δ equals 0, the queried DTs must be synchronized at the current time slot, only 5.97% evaluation plans among 1,000 query plans are feasible. The percentage of feasible evaluation plans can reach 99.32% when δ is set as 4. We also studied the impact of the data fresh fidelity parameter ϑ on the query result for a query, by increasing its value from 0.2 to 1.0. It can be seen from Fig. 3.7 (b) the percentage of feasible evaluation plans becomes decreasing, with the increase of ϑ. The reason is that fewer numbers of DTs involving the queries can meet the data fresh fidelity requirements.

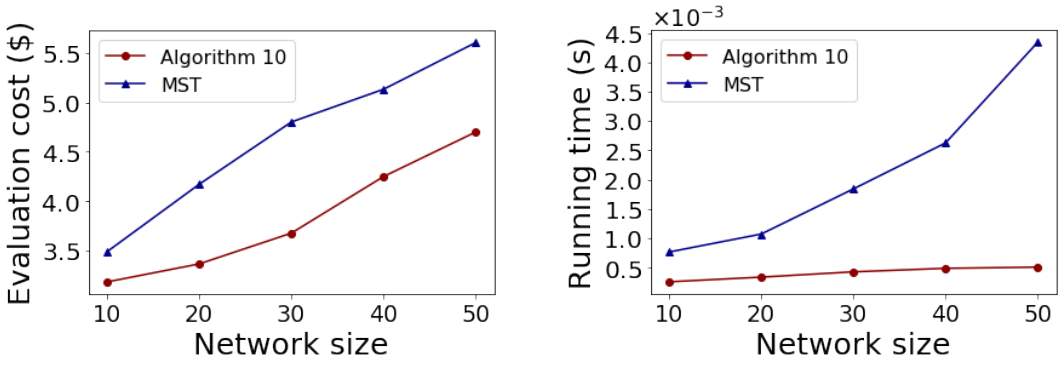


(a) (b)

Figure 3.7: The percentage of feasible evaluation plans from 1,000 queries randomly generated by varying δ and varying ϑ respectively.

We then investigated the impact parameters: the data fresh fidelity parameter ϑ, the number of querying sensors $|V'|$, the network size $|\mathcal{N}|$, the transmission cost per unit data ρ, and the computing cost per unit data ξ, on the cost of query evaluation plans. The result is the mean result of 1,000 feasible queries.

We evaluated the impact of network size $|\mathcal{N}|$ by varying its value from 10 to 50. It can be seen from Fig. 3.8 (a) that the evaluation cost increases with the increase on network size. This can be justified by the growth of the transmission cost of DT information is due to the increase on network size. The evaluation cost by algorithm `MST` is much higher than that by `Algorithm` 10. The evaluation tree delivered by algorithm `MST` does not lead to the minimum transmission cost. Moreover, algorithm `MST` does not consider the computing cost of aggregation at cloudlets. Fig. 3.8 (b)

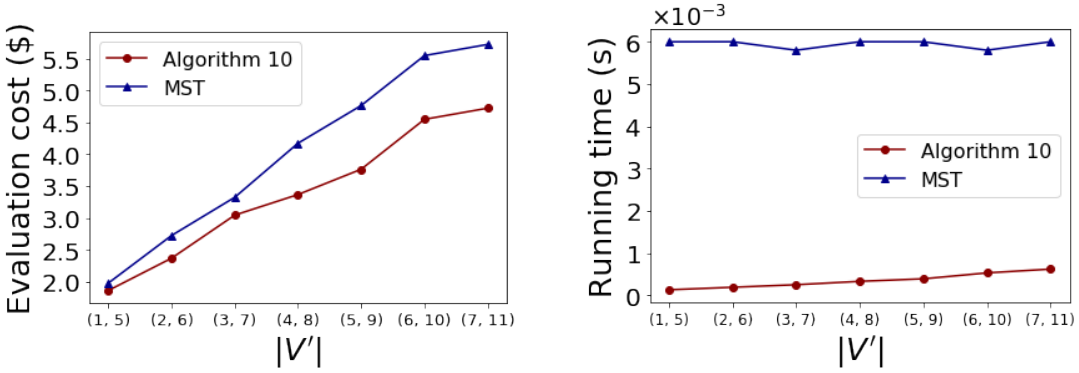


(a) Evaluation cost (b) Average running time

Figure 3.8: Performance of different algorithms for the fidelity-aware query evaluation problem by varying network size $|\mathcal{N}|$.

depicts the running times of different algorithms.

We investigated the impact of the number $|V'|$ of sensors requested per query, by varying its value range from $[1,5]$ to $[7,11]$. Fig. 3.9 (a) shows that the evaluation cost increases with the growth of $|V'|$. This is due to additional computing and transmission costs incurred by requesting the DT information from more sensors. Fig. 3.9 (b) depicts the running times of the comparison algorithms. The running time of `Algorithm` 10 increases insignificantly as more iterations of adding cloudlets to the partial evaluation tree are needed, while the running time of algorithm `MST` does not change too much because finding a minimum spanning tree only depends on the number of edges in the network.

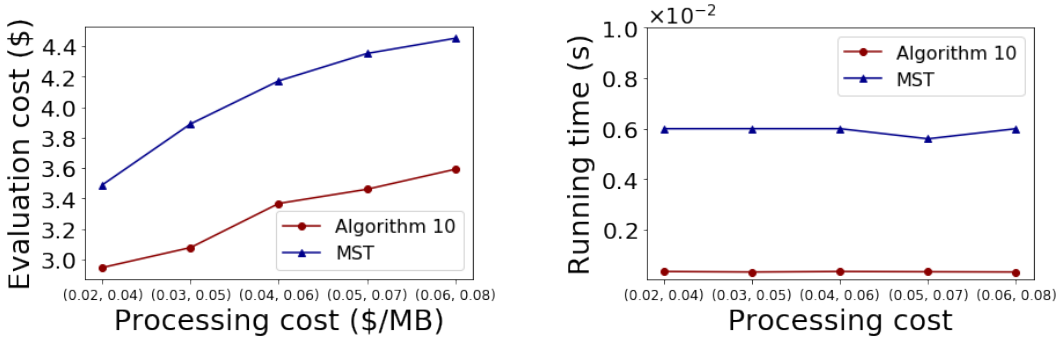


(a) Evaluation cost (b) Average running time

Figure 3.9: Performance of different algorithms for the fidelity-aware query evaluation problem by varying the number of querying sensors $|V'|$.

We finally studied the impact of the processing cost per unit data ξ by varying its value range from $[\$0.02, \$0.04]$ to $[\$0.06, \$0.08]$. Fig. 3.10 (a) depicts that both algorithms deliver evaluation trees with higher evaluation costs. When the processing cost is drawn from $[0.06, 0.08]$, the cost of the evaluation tree delivered by `Algorithm` 10 is 24.01% higher than that when the processing cost from $[0.02, 0.04]$.

Fig. 3.10 (b) shows that the running times of both algorithms remains unchanged with the increase of processing cost.



(a) Evaluation cost

(b) Average running time

Figure 3.10: Performance of different algorithms for the fidelity-aware query evaluation problem, by varying the processing cost per unit data ξ.

3.7 Conclusion

In this chapter, we considered the state synchronization issue between digital twins and their physical objects (sensors) for an application scenario: a UAV-enabled data collection in a renewable sensor network. We first formulated a novel staleness minimization problem of DT states. We then proposed an optimal algorithm for a special case of the problem when the update budget is exactly the DT state updating of K sensors per update round. We also devised a heuristic algorithm for the problem by reducing it to the award collection maximization problem under the assumption that the volume of data generated by each sensor from its last data collection to now is given. To remove this assumption, we adopted the LSTM method - a deep learning mechanism to predict the volume of data generated by each sensor, by analyzing the historical traces of the data generated by the sensor through its DT. To demonstrate the importance of maintaining the average staleness of DT states of objects, we thirdly proposed a query evaluation algorithm for fidelity-aware queries built upon the DT data. We finally evaluated the performance of the proposed algorithms and the prediction mechanism through simulations. Simulation results demonstrate that the proposed algorithms are promising.

Chapter 4

Profit Driven Service Provisioning via Deep Reinforcement Learning

In this chapter, we study on the online service placement and request assignment problem with the aim to maximize the profit of the service provider. This optimization objective is achieved by assigning service requests to appropriate cloudlets in the MEC network, pre-installing service instances into cloudlets to shorten service delays, and accommodating new services by revoking some idle service instances from cloudlets due to limited computing resources in MEC networks. We first devise an efficient deep reinforcement learning algorithm for the online service placement and request assignment problem that consists of a deep reinforcement learning-based prediction mechanism for dynamic service placement, followed by a dynamic request assignment procedure to assign requests to cloudlets.

4.1 Introduction

With the integration of Mobile Edge Computing (MEC) and Network Function Virtualization (NFV), service providers are able to provide low-latency services to mobile users for profits. When a user requests a service, its service provider will admit the request and initialize a service instance for the user. Once the requested service finishes, the service instance will be released back to the system to reduce the operational cost of the service provider, and the released resources can be utilized by future service requests. Most existing studies focused on how to assign service requests to different cloudlets in an MEC network, by either instantiating new service instances or sharing existing service instances [60, 61, 104, 128]. In contrast, in this chapter we consider a scenario where the service provider of an MEC network can pre-install most demanded service instances in cloudlets in advance. The pre-installed service instances can be used by later service requests immediately without service initialization delays, this can shorten service delays of service requests. Consequently, the service provider can earn more profits by admitting more service requests while satisfying their service delay requirements. Since the service provider can only place limited service instances at resource-constrained cloudlets, an effective prediction mechanism that can predict which types of service requests

and how many of the service instances needed in a foreseeable future.

The novelties of this chapter lie in a framework for dynamic service placement, by pre-installing service instances and revoking idle service instances through an effective prediction mechanism, with the aim to maximize the profit of the service provider. A deep-reinforcement learning-based online algorithm is devised through learning historical traces of service request patterns and service types at different cloudlets. A solution that can provide services for various requests then is delivered.

The rest of the chapter is organized as follows. Section 4.2 introduces the system model and problem definition. Section 4.3 develops an efficient algorithm for dynamic service request admissions that consists of a deep reinforcement learning-based prediction mechanism to predict different types of service request arrivals in order to install their service instances in the system, and an online algorithm for request assignment based on the installed service instances in cloudlets. Section 4.4 evaluates the performance of the proposed algorithms empirically, and Section 4.5 concludes the chapter.

4.2 Preliminaries

In this section, we introduce the system model and define the problem precisely.

4.2.1 System model

We consider a mobile edge computing network $\mathcal{G} = (\mathcal{V} \cup S, E)$ that consists of a set $\mathcal{V}$ of access points (APs) and a set E of links between APs. For each link $e(i,j) \in E$, there is a transmission delay l_e $(= l_{i,j})$ on it. We assume that a link between two APs is an optical link with high capacity, and thus the bandwidth on the link is not considered as a constraint. Also, a set S of cloudlets are co-located with some of the APs. Each cloudlet $c_j \in S$ has limited computing capacity CAP_j for hosting service instances. Let $f_k \in F$ with $k = 1, 2, \ldots, |F|$ be a service function in the service set F provided by the MEC network. We assume that a service instance of f_k can only serve a request at any moment, and thus, the computing resource requirement of a service instance can be precisely estimated in advance. Denote by $c(f_k)$ the computation resource requirement of service instance f_k. We further assume that the monitoring time period T is divided into equal time slots indexed by t with $1 \leq t \leq T$. In order to reduce the end-to-end service delay of each service request, service instances can be pre-installed in cloudlets prior to arrivals of requests. Meanwhile, some idle service instances can also be removed from the system to deploy the most demanded services. Therefore, each time slot is split into two stages: *the service placement* stage, and *the request admission* stage. Fig. 4.1 demonstrates an MEC network with five APs and three co-located cloudlets, where several service instances are installed at cloudlet 1 and cloudlet 3, respectively.

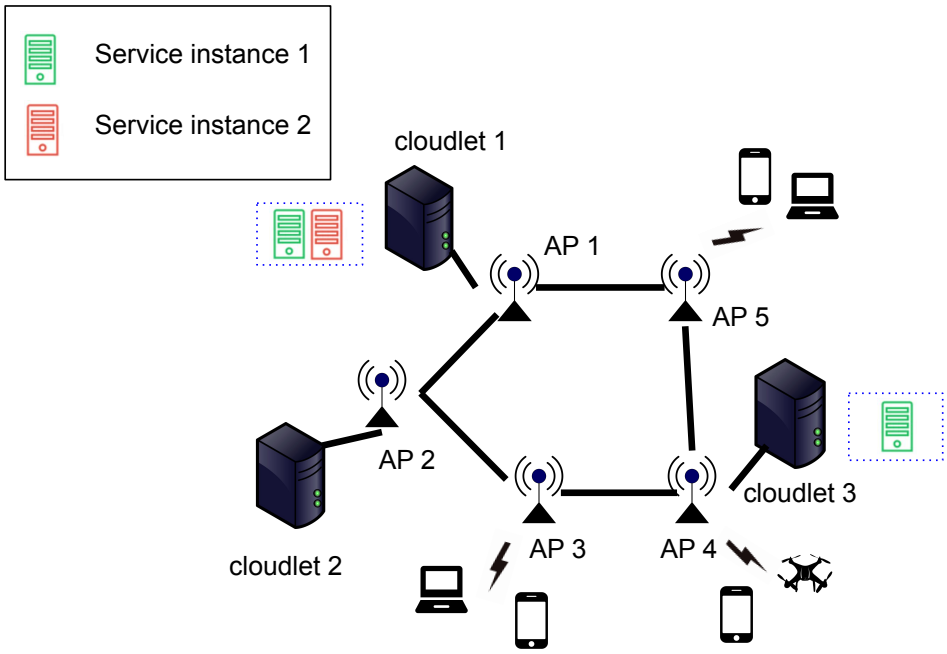


Figure 4.1: An illustration of an MEC network with installed service instances.

4.2.2 Service placement stage

Denote by $n_{j,k}^t$ the number of instances of service f_k in cloudlet c_j at time slot t with $1 \leq t \leq T$. Since each cloudlet has limited computing capacity, the number of service instances installed in cloudlet c_j at time slot t is constrained by $\sum_{k=1}^{|F|} n_{j,k}^t \leq CAP_j$, where CAP_j is the computing capacity of c_j. As service instances in cloudlets at each time slot t can be added or removed, let $\delta_{j,k}^t$ be the number of service instances to be installed or deleted from cloudlet c_j at time slot t. If $\delta_{j,k}^t > 0$, then cloudlet c_j instantiates $\delta_{j,k}^t$ instances of service f_k; otherwise, $\delta_{j,k}^t$ instances of f_k will be released from c_j. Furthermore, considering that some instances are serving unfinished requests, they should not be removed. Thus, the number of instances at each cloudlet should be no less than the number of unfinished requests at the cloudlet. For convenience, denote by $\phi_{j,k}^t$ the number of requests that are still being processed at time slot t, we have $\phi_{j,k}^t \leq n_{j,k}^t$.

4.2.3 Request admission stage

Mobile users offload their tasks to cloudlets via their nearby APs. Let R be the set of service requests arrived at different APs during a monitoring period of T. We assume that at time slot t, a subset of requests $R(t)$ ($\subset R$) arrives. Request $r_i \in R$ can be represented by a tuple $(t_i, f_{k_i}, v_i, \tau_i, d_i)$, where t_i is the time slot of its arrival; $f_{k_i} \in F$ represents its service type, $v_i \in \mathcal{V}$ represents its closest AP, τ_i represents its execution

duration, and d_i is its end-to-end service delay requirement. For convenience, denote by $\mathcal{F}(k)$ ($\subset R$) the set of requests that demand service f_k.

Given an arrived request r_i, it will be either admitted by assigning it to one cloudlet or rejected immediately. We use a binary variable $x_{i,j}^t \in \{0,1\}$ to denote whether request r_i is admitted by cloudlet c_j at time slot t. Request r_i can only be admitted at the time slot of its arrival,

$$x_{i,j}^t = 0, \quad \forall i,j,t \neq t_i. \tag{4.1}$$

Because each service request r_i has an execution duration of τ_i time slots, once it is admitted, the request leaves the MEC system at the end of the $(t_i + \tau_i - 1)th$ time slot. We use $P_{i,j}^t \in \{0,1\}$ to denote that request r_i is being processed by cloudlet c_j at time slot t, and thus

$$\forall i,j,t \quad P_{i,j}^t = \begin{cases} x_{i,j}^{t_i}, & t_i \leq t < t_i + \tau_i \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

To admit request r_i, a cloudlet c_j needs to have an idle service instance or sufficient resource to initialize a new service instance for f_{k_i}. For convenience, we use a binary variable $x_{t,i,j}^{idle} \in \{0,1\}$ to indicate if r_i is offloaded to an idle instance, and $x_{i,j}^{t^{new}} \in \{0,1\}$ to indicate if request r_i is served by initializing a new instance. We then have

$$x_{i,j}^t = x_{t,i,j}^{idle} + x_{t,i,j}^{new}, \quad \forall i,j,t. \tag{4.3}$$

Recall that $\phi_{j,k}^t$ is the number of unfinished requests for service f_k at the service placement stage of time slot t, we have

$$\phi_{j,k}^t = \sum_{r_i \in \mathcal{F}(k) \cap R(t)} P_{i,j}^t, \quad \forall t,j,k. \tag{4.4}$$

Also, if there are $n_{j,k}^t$ instances for service f_k in cloudlet c_j at time slot t before the service admission stage of time slot t and $x_{t,i,j}^{new}$ instances are created at the request admission stage, the number of instances of service f_k at time slot $t+1$ then is

$$n_{j,k}^{t+1} = n_{j,k}^t + \sum_{r_i \in \mathcal{F}(k)} x_{t,i,j}^{new} + \delta_{j,k}^{t+1}, \quad \forall t,j,k. \tag{4.5}$$

To admit a request, not only does the cloudlet have sufficient residual computation capacity, but also the delay requirement d_i of request r_i can be met. Assigning request r_i to cloudlet c_j will experience the network delay, potential initialization delay, and processing delay. Denote by $proc_k$ the processing delay of an instance for service f_k, and define the network delay as the accumulative delay along the shortest path from the arrived AP to the target cloudlet, where a target cloudlet of a request is referred to its assignment to that cloudlet. Recall that link $e \in E$ between two APs has a latency l_e. The network delay of request r_i from an AP v_i to cloudlet c_j

is $L_{i,j} = \sum_{e \in p_{i,j}} l_e$, where $p_{i,j}$ is the shortest path in the MEC network between AP v_i and AP v_j. We assume that the initialization time of service f_k is ins_k. If request r_i is uploaded to cloudlet c_j, and c_j does not have an idle instance for f_{k_i}, the cloudlet will instantiate a new instance for f_k. This will result in an initialization delay. The latency of assigning request r_i to cloudlet c_j thus is $L_{i,j} + ins_{k_i}$, which should be no greater than the delay requirement of request r_i, i.e.,

$$(L_{i,j} + proc_{k_i}) \cdot x_{i,j}^t + x_{t,i,j}^{new} \cdot ins_{k_i} \leq d_i, \quad \forall i, j, t. \tag{4.6}$$

4.2.4 Operational expense and request admission profit

Initializing and hosting a service instance at any cloudlet incurs an operational cost. The operational expenditure of an MEC network consists of three components: (i) the energy consumption for hosting service instances even if the service instances are idle; (ii) the energy consumption for serving each service request; and (iii) the initialization cost of service instances.

Let $\mathcal{C}^{idle}(f_k)$ be the energy consumption per time slot for hosting an instance of service f_k, $\mathcal{C}^{ser}(f_k)$ the energy cost per time slot for admitting a request of service f_k, and $\mathcal{C}^{ins}(f_k)$ the initialization cost of an instance of service f_k. The operational cost $\mathcal{C}_j(t)$ of cloudlet c_j at time slot t thus is

$$\begin{aligned}\mathcal{C}_j(t) = \sum_{k=1}^{|F|}\Big(\Big(\sum_{r_i \in R(t) \cap \mathcal{F}(k)} x_{t,i,j}^{new} + [\delta_{j,k}^t]^+\Big) \cdot \mathcal{C}^{ins}(f_k) + \Big(\sum_{r_i \in R(t) \cap \mathcal{F}(k)} x_{t,i,j}^{new} + n_{j,k}^t\Big) \cdot \mathcal{C}^{idle}(f_k) \\ + \sum_{r_i \in R(t) \cap \mathcal{F}(k)} P_{i,j}^t \cdot \mathcal{C}^{ser}(f_k)\Big),\end{aligned} \tag{4.7}$$

where $[a]^+ = \max(a, 0)$, the first term in Eq (4.7) is the cost of initializing new instances, the second term is the idle energy consumption cost of hosting instances, and the third term is the energy consumption of serving requests.

The service provider charges users for their service requests are admitted. We assume that the service provider will earn the amount $RV(f_k)$ of revenues per time slot for serving a request of service f_k. Then, the total revenue obtained by cloudlet c_j at time slot t is

$$RV_j(t) = \sum_{r_i \in R(t)} \tau_i \cdot RV(f_{k_i}). \tag{4.8}$$

The profit of cloudlet c_j at time slot t is $RV_j(t) - \mathcal{C}_j(t)$. The total profit of the service provider by admitting user service requests for a given monitoring period T thus is

$$\sum_{t=1}^{T} \sum_{j=1}^{|S|} (RV_j(t) - \mathcal{C}_j(t)). \tag{4.9}$$

4.2.5 Problem formulation

The *online service placement and request assignment* problem is defined as follows. Given a mobile edge-cloud network $\mathcal{G} = (\mathcal{V} \cup S; E)$, a set F of services provided by the network, a finite time horizon that is divided into T equal time slots, and a sequence of service requests arriving one by one without the knowledge of future arrivals, let R be the set of arrived requests within the given time horizon T, where each request $r_i \in R$ demands a service $f_{k_i} \in F$ with a delay requirement d_i. The problem is to assign arrived requests at each time slot t to cloudlets while meeting the delay requirements of admitted requests through (i) pre-deploying service instances of services in F to different cloudlets and determining numbers of service instances to meet the service delays of admitted requests; and (ii) removing some existing idle service instances from the system such that the accumulative profit $\sum_{j=1}^{|S|} \sum_{t=1}^{T} (RV_j(t) - \mathcal{C}_j(t))$ is maximized, subject to computing capacity on each cloudlet in $\mathcal{G}$.

Theorem 4.1. *The online service placement and request assignment problem in $\mathcal{G} = (\mathcal{V} \cup S, E)$ is NP-hard.*

Proof. We prove that the one-time-slot service placement problem with the knowledge of arrived requests is NP-hard, by a reduction from a well-known NP-hard problem - the multi-knapsack problem as follows.

Given a set B of knapsacks, where each knapsack $b_j \in B$ has capacity $W(b_j)$, and a set A of items where each item $a_i \in A$ has value $value(a_i)$ and size $size(a_i)$. The multi-knapsack problem is to maximize the accumulative value by placing as many items as possible into knapsacks, subject to the capacity of each knapsack.

Given an instance of a multi-knapsack problem, we construct a special case of the online service placement and request assignment problem as follows. For each knapsack b_j, we create a cloudlet c_j with capacity $W(b_j)$, and there is no idle service instance at the cloudlet. For each item a_i, we construct a request r_i with one-time-slot duration and the profit to admit it is $value(a_i)$, and the service instance f_{k_i} needs $size(a_i)$ computation capacity. Here we assume that the delay requirement is stringent that the requests can be admitted only if their required service instances are ready, and we also assume that the network delay is negligible so there is no difference among cloudlets when admitting requests. We aim to find how many instances to be installed at cloudlets while the computing capacity on each cloudlet is constrained. It can be seen that a solution to this special service placement and request assignment problem is a solution to the multi-knapsack problem, and the reduction is polynomial. Moreover, the online service placement and request assignment problem needs to place the instances without knowing the arrived requests at the current time slot, considering the impact of placement on future arrived requests, which is at least as hard as the special service placement and request assignment problem. Therefore, the problem is NP-hard. □

4.2.6 ILP formulation of the offline version of the problem

In the following, we formulate an ILP formulation for the offline service placement and request assignment problem, where the set $R(t)$ of requests at each time slot t is given with $1 \leq t \leq T$, i.e., $R = \cup_{t=1}^{T} R(t)$. For this offline version, the objective is to maximize the accumulative profit as follows.

$$\underset{x^{new}_{t,i,j}, x^{idle}_{t,i,j}, \delta^t_{j,k}}{\text{Maximize}} \quad \sum_{t=1}^{T} \sum_{j=1}^{|S|} (RV_j(t) - \mathcal{C}_j(t)) \tag{4.10}$$

$$s.t. \quad (4.1), (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9)$$

$$\sum_{k=1}^{|F|} n^t_{j,k} \cdot c(f_k) \leq CAP_j, \quad \forall j, t \tag{4.11}$$

$$\sum_{k=1}^{|F|} n^t_{j,k} \cdot c(f_k) + \sum_{i \in R(t)} x^{new}_{t,i,j} \cdot c(f_{k_i}) \leq CAP_j, \quad \forall j, t \tag{4.12}$$

$$\sum_{i \in \mathcal{F}(k)} x^{idle}_{t,i,j} \leq n^t_{j,k} - \phi^t_{j,k}, \quad \forall j, k, t \tag{4.13}$$

$$\phi^t_{j,k} \leq n^t_{j,k}, \quad \forall j, k, t \tag{4.14}$$

$$\sum_{t=1}^{T} \sum_{j=1}^{|S|} x^t_{i,j} \leq 1, \quad \forall i \tag{4.15}$$

$$n^t_{j,k} \geq 0, \quad \forall j, k, t \tag{4.16}$$

$$\delta^t_{j,k} \in Z, \quad \forall j, k, t \tag{4.17}$$

$$x^t_{i,j} \in \{0, 1\}, \quad \forall i, j, t \tag{4.18}$$

$$x^{idle}_{t,i,j} \in \{0, 1\}, \quad \forall i, j, t \tag{4.19}$$

$$x^{new}_{t,i,j} \in \{0, 1\}, \quad \forall i, j, t \tag{4.20}$$

Constraint (4.11) guarantees that the total computing resource demand of all instances in the service placement stage does not exceed the computing capacity on each cloudlet at any time slot. Constraint (4.12) prevents cloudlets from admitting too many requests to violate their computation capacity. The RHS of Constraint (4.13) denotes the number of idle service instances. The LHS in Constraint (4.13) represents the number of requests for service f_k admitted by idle instances at cloudlet c_j, which should be no larger than the RHS. Constraint (4.14) ensures that the number of service instances is no less than the number of instances that are being processed, so in service placement stage we do not remove non-idle service instances. Constraint (4.15) ensures that each request can only be admitted by one cloudlet.

Although the ILP formulation of the offline version of the problem can deliver an optimal solution, this solution is an upper bound on the problem due to that it is assumed that the knowledge of request arrivals at all time slots is given. The LP

relaxation of the proposed ILP solution delivers a solution whose value is an upper bound on the value of the optimal solution of the offline version of the problem, which is also an upper bound on the problem. This relaxed solution thus will be used as an estimate of the optimal solution of the problem later.

4.3 Online Algorithm

In this section we present an online algorithm for the online service instance placement and request assignment problem.

4.3.1 Algorithm overview

The basic idea of the proposed algorithm consists of two stages: the service placement stage, followed by the request assignment stage. We first adopt a deep reinforcement learning method to predict service request demands and types of services. We then find a near-optimal service placement for the predicted service instances. We finally assign each incoming service request to a cloudlet that contains the service instance or instantiate an instance of the demanded service.

4.3.2 Request assignment

Assuming that service instances have been placed in cloudlets, we deal with service request assignment with the aim to maximize the accumulative profit at each time slot t with $1 \leq t \leq T$. Given that a set $R(t)$ of requests arrives and some service instances have already been deployed in cloudlets at time slot t, we reduce request assignment to the Generalized Assignment Problem (GAP) [27] to maximize the profit.

Given a one-time-slot request assignment, we treat each idle service instance of f_k as a bin with capacity $c(f_k)$. Also, each cloudlet is treated as a bin with its remaining computing capacity $CAP_j - \sum_{k=1}^{|F|} n_{j,k}^t \cdot c(f_k)$ at time slot t. Then, each request r_i corresponds to an item. For bins corresponding to idle service instances, if the delay requirement of a request can be satisfied when assigned to the bin, its size is set as $c(f_{k_i})$, and the profit is set as $(RV(f_{k_i}) - \mathcal{C}^{ser}(f_{k_i})) \cdot \tau_i$. Similarly, for bins corresponding to the residual computing capacities of cloudlets, the size of an item at the bins is $c(f_{k_i})$, and the profit is set as $(RV(f_{k_i}) - \mathcal{C}^{ser}(f_{k_i}) - \mathcal{C}^{idle}(f_{k_i})) \cdot \tau_i - \mathcal{C}^{ins}(f_{k_i})$ due to the extra cost introduced by initializing new instances. For idle service instances or cloudlets that cannot fulfil the delay requirement of request r_i, we set its size as ∞ to indicate that r_i cannot be admitted. It can be seen that an approximate solution to the GAP in turn returns an approximate solution to the request assignment problem. The detailed procedure is given in `Procedure` 1.

Procedure 1 Request assignment procedure at one time slot

Input: An MEC network $\mathcal{G} = (\mathcal{V} \cup S; E)$ with a set of cloudlets C, the number of total service instance $n_{j,k}^t$ and occupied service instance $\phi_{j,k}^t$ for service f_k at each cloudlet c_j, and a set of requests $R(t)$ that arrives at time slot t.

Output: Find an assignment of requests in $R(t)$ at time slot t so that the total profit is maximized, subject to the capacity of cloudlets and delay requirement.

Calculate the delay L between any pairs of APs and APs co-located with cloudlets by the Floyd-Warshall algorithm [28];
Initialize an empty set of bins $\mathcal{B} \leftarrow \emptyset$;
for $j \leftarrow 1, 2, \ldots, |S|$ **do**
    for $k \leftarrow 1, 2, \ldots, |F|$ **do**
        for $b \leftarrow 1, 2, \ldots, n_{j,k}^t - \phi_{j,k}^t$ **do**
            Create a new bin B with capacity $c(f_k)$;
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{B\}$;
    Create a new bin B' with capacity $CAP_j - \sum_{k=1}^{|F|} n_{j,k}^t \cdot c(f_k)$;
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{B'\}$;
Initialize an empty set of items $\mathcal{I} \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, |R(t)|$ **do**
    Create a new item I and $\mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$;
    for $j' \leftarrow 1, 2, \ldots, |\mathcal{B}|$ **do**
        if Bin B_j' corresponds to an idle service instance at cloudlet c_j **then**
            $Profit(i, j') \leftarrow (RV(f_{k_i}) - \mathcal{C}^{ser}(f_{k_i})) \cdot \tau_i$;
            $delay(i, j') \leftarrow L_{i,j}$;
        else
            $Profit(i, j') \leftarrow (RV(f_{k_i}) - \mathcal{C}^{ser}(f_{k_i}) - \mathcal{C}^{idle}(f_{k_i})) \cdot \tau_i - \mathcal{C}^{ins}(f_{k_i})$;
            $delay(i, j') \leftarrow L_{i,j} + ins_{k_i}$;
        if $delay(i, j') \leq D_i$ **then**
            $weight(i, j') \leftarrow c(f_{k_i})$;
        else
            $weight(i, j') \leftarrow \infty$;
Find an assignment of items $\mathcal{I}$ to $\mathcal{B}$ such that the accumulative profit of assigned items is maximized under the constraint that the total weight of items in each bin does not exceed its capacity, by invoking the approximation algorithm for the GAP in [27];
return The assignment of requests $x_{t,i,j}^{new}$ and $x_{t,i,j}^{idle}$ derived from the assignment of items.

4.3.3 Service placement

Considering that no prior knowledge of request arrivals at each time slot t is given, we adopt a deep reinforcement learning method for service placement. Specifically, we first formulate the service placement as a Multi-agent Markov Decision Process (MMDP), where a set of agents interacts with an environment for a given number of time slots, with the aim to maximize the total reward. At each time slot, the agents obtain the state of the environment, and then take actions based on the

state. The actions then impact the environment and transit the environment to the next state. The agents receive rewards as a consequence of their actions and the change of the environment. Fig. 4.2 demonstrates the interaction of agents and their environments.

An MMDP is usually expressed by a tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{N}$ is the set of agents, $\mathcal{S}$ is the state set of the environment, $\mathcal{A}$ is the action set of agents, $\mathcal{P}$ is the transition function, and $\mathcal{R}$ is the reward function. Given state s, actions $a_1, a_2, \ldots, a_{|\mathcal{N}|}$ and the next state s', $\mathcal{P}(s, a_1, a_2, \ldots, a_{|\mathcal{N}|}, s')$ defines the probability of the transition from state s to s'. Similarly, $\mathcal{R}(s, a_1, a_2, \ldots, a_{|\mathcal{N}|}, s')$ is the reward that the agents obtain by taking action a when the state transits from s to s'. We formulate the problem as

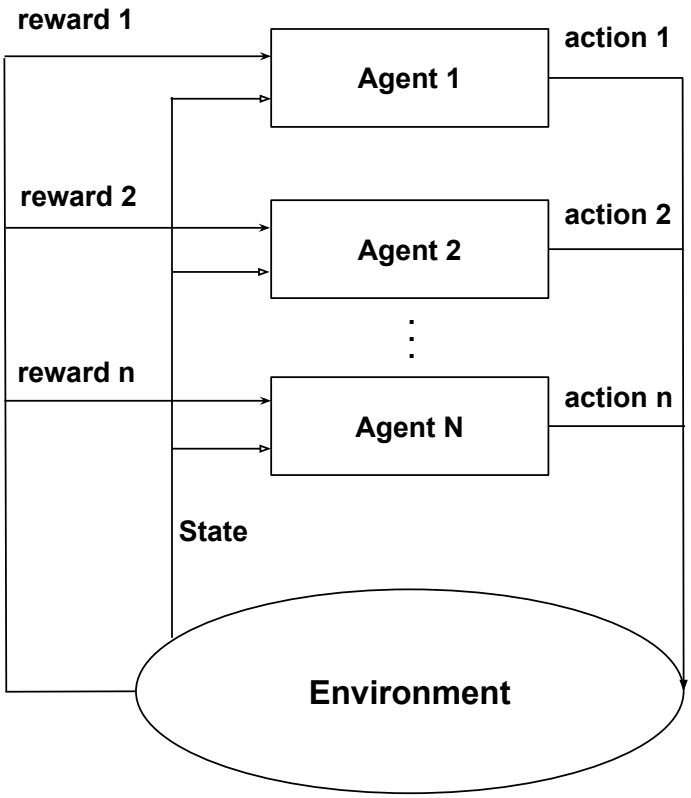


Figure 4.2: An illustration of a Multi-agent Markov Decision Process in one time slot.

a multi-agent Markov Decision Process (MMDP) as follows.

Agent: Each cloudlet c_j with $j \in \{1, 2, \ldots, |S|\}$ is modeled by an agent $\mathcal{N}_j$ to manage its service placement.

State: The state $s \in \mathcal{S}$ describes the status of the MEC network, i.e., the history of offloaded service requests that will be admitted at cloudlets and their demanded service instances in cloudlets. Thus, the system state $s^t \in \mathcal{S}$ at time slot t is defined by a tuple $(R_\eta(t), n(t), R^*(t), t)$, where $R_\eta(t) = R(1) \cup R(2) \cup \ldots \cup R(t-1)$ is the set

of service requests that arrive before time slot t, $n^t_{j,k} \in n(t)$ is the number of instances of service k at cloudlet c_j, $R^*(t)$ denotes the set of admitted service requests prior to time slot t, that have not finished at time slot t. **Action:** In the service placement stage of each time slot, the agents decide to place or remove some service instances at their cloudlets. Thus, action $a_j \in \mathcal{A}$ of agent $\mathcal{N}_j$ is defined as a vector $a_j = (a_{j,0}, a_{j,1}, a_{j,2}, \ldots, a_{j,|F|})$, $\sum_{k=0}^{|F|} a_{j,k} = 1$, where $a_{j,k}$ represents the percentage of available computational capacity to be allocated to service f_k if $k \neq 0$ and $a_{j,0}$ represents the percentage of available computational capacity. While deciding the placement of instances, we need to maintain the instances of unfinished requests. Therefore, the available computational capacity that can be allocated is $cap^{rej}_{t,j} = CAP_j - \sum_{k=1}^{|F|} \phi^t_{j,k} \cdot c(f_k)$. The action a_j can be decoded by $\delta^t_{j,k} = \lfloor \frac{cap^{rej}_{t,j} \cdot a'_{j,k}}{c(f_k)} \rfloor$ for service placement.

State Transition: Having taken agent actions, the system state will transit to the next state, following the state transition function $P(s^t, a^t_1, a^t_2, \ldots, a^t_{|S|}, s^{t+1})$. Specifically, first, the number of instances at any cloudlet $c_j \in S$ changes under the agent's action a_j in the service placement stage, that is, $n^t_{j,k} \leftarrow n^t_{j,k} + \delta^t_{j,k}$. Second, in the request admission stage, a set of requests $R(t)$ arrives. `Procedure` 1 (that will be discussed later) is applied to assign the arrived requests to cloudlets. The system state then is updated, following the result delivered by `Procedure` 1. Since finished requests will leave in the end of each time slot, set $R^*(t)$ will be updated accordingly.

Reward: To encourage each agent to maximize its contribution to the total profit, a reward function $\mathcal{R}_j(s^t, a^t_1, a^t_2, \ldots, a^t_{|S|}, s^{t+1})$ is defined, which describes how much reward an agent $\mathcal{N}_j$ receives after a state transition. Assume that there is a state transition $s^t = (R_\eta(t), n(t), R^*(t), t) \xrightarrow{a^t_1, a^t_2, \ldots, a^t_{|S|}} s^{t+1} = (R_\eta(t+1), n(t+1), R^*(t+1), t+1)$. Let $x^{new}_{t,i,j}$ and $x^{idle}_{t,i,j}$ be the outputs of `Procedure` 1 during the transition. The revenue and operation cost of cloudlet c_j are calculated by Eq. (4.7) and Eq. (4.8), respectively.

The reward function of agent $\mathcal{N}_j$ then is defined as follows.

$$\mathcal{R}_j(s^t, a^t_1, a^t_2, \ldots, a^t_{|S|}, s^{t+1}) = RV_j(t) - \mathcal{C}_j(t). \tag{4.21}$$

The cumulative reward of agent $\mathcal{N}_j$ from time slot t' to T then is

$$G_j(t') = \sum_{t=t'}^{|T|} \mathcal{R}_j(s^t, a^t_1, a^t_2, \ldots a^t_{|S|}, s^{t+1}). \tag{4.22}$$

It can be seen that $G_j(1)$ is the total reward of agent $\mathcal{N}_j$ during the monitoring period T, so $\sum_{j=1}^{|S|} G_j(1)$ is equivalent to the objective function (4.9). We thus maximize the total reward $G_j(1)$ of all agents $\mathcal{N}_j$ instead of maximizing the original optimization objective of the problem. Due to the large state space and lack of knowledge of future request arrivals, it is impossible to calculate the state transition function and reward function accurately. We instead design a policy function $\pi_j(a_j \mid s)$ for each agent $\mathcal{N}_j$, which gives the possibility of taking action a under state s. We then find an optimal

policy function of each agent. We make use of deep neural networks (DNNs) called *actor networks* as the policy function $\pi_j(a_j \mid s)$, which can extract important features from states and learn service request patterns. The service placement by DNNs can be obtained by applying `Procedure` 2 as follows.

Procedure 2 Advantage Actor-Critic based procedure for online service instance placement

Input: The system state $s^t = (R_\eta(t), n(t), R^*(t), t)$, the actor network with parameter θ_j for agent $\mathcal{N}_j$.

Output: The service placement decision for cloudlet c_j

1: Let $\pi_j(a_j^t \mid s^t; \theta_j)$ be the output of the actor network for the input of system state s^t;

2: Sample action a_j^t according to the distribution $\pi_j(a_j^t \mid s^t; \theta_j)$;

3: $\delta_{j,k}^t \leftarrow \lfloor \frac{cap_{t,j}^{rej} \cdot a_{j,k}'^t}{c(f_k)} \rfloor$, $\forall k \in \{1, 2, \ldots, |F|\}$; **return** The service placement decision $\delta_{j,k}^t$ for cloudlet c_j

Initially, each agent in the actor network is not trained and has a random parameter θ_j. The reward could be low. To obtain a near-optimal reward, assume that the service provider has a collection Γ of sets of historical requests in the past monitoring periods, where a set $R' \in \Gamma$ of requests arrives during one of the past monitoring periods which has an almost identical arriving pattern as R. We train the actor networks using historical data traces to approximate the optimal policy functions of agents as disscussed below.

4.3.4 Online algorithm

Considering that there are multiple agents and the action space is continuous. To train the actor networks, we adopt the Advantage Actor-Critic (A2C) algorithm for multiple agents and continuous space scenario [132], where the actor networks of all agents have randomly initialized parameter θ_j. Denote by $\pi_j(a_j \mid s^t; \theta_j)$ the actor network with the parameter θ_j for each agent $\mathcal{N}_j$.

The basic idea of the A2C algorithm is to apply Stochastic gradient descent(SGD) technique [55] to update parameter θ_j in order to maximize the expected cumulative reward of agent $\mathcal{N}_j$. The expected cumulative reward of agent $\mathcal{N}_j$ from time slot 1 to T is defined as a function J_j^t with respect to parameter θ_j by

$$J_j(\theta_j) = \mathbb{E}[G_j(1); \theta_j]. \tag{4.23}$$

Since we aim to maximize $J_j(\theta_j)$, this is equal to minimizing $-J_j(\theta_j)$. We update parameters θ_j of the actor network by applying SGD, i.e.,

$$\theta_j \leftarrow \theta_j + \eta \nabla_{\theta_j} J_j(\theta_j), \tag{4.24}$$

where $\nabla_{\theta_j} J_j(\theta_j)$ is the gradient of $J_j(\theta_j)$, and η is the learning rate to control the step of the gradient descent, which is a constant with $0 < \eta < 1$.

Denote by $\zeta = (s^1, a_1^1, ..., a_{|S|}^1, s^2, ..., a_{|S|}^{|T|}, s^{|T|})$ a sequence of system states and agents' actions for the monitoring period T. Let $Pr(\zeta)$ be the probability of sequence ζ and $\mathcal{R}_j(\zeta)$ the reward that agent $\mathcal{N}_j$ obtained if the sequence of system states and agent actions follows ζ. Eq.(4.23) can be written as

$$J_j(\theta_j) = \int Pr(\zeta) \cdot \mathcal{R}_j(\zeta) d\zeta. \tag{4.25}$$

Note that for function $f(\cdot)$, we have $\nabla f(\cdot) = f(\cdot)\nabla \log f(\cdot)$. To calculate the gradient $\nabla_{\theta_j} J_j(\theta_j)$, we have

$$\begin{aligned}
\nabla_{\theta_j} J_j(\theta_j) &= \nabla_{\theta_j} (\int Pr(\zeta) \cdot \mathcal{R}_j(\zeta)\ d\zeta) \\
&= \int \mathcal{R}_j(\zeta) \cdot (\nabla_{\theta_j} Pr(\zeta))\ d\zeta \\
&= \int Pr(\zeta) \cdot \mathcal{R}_j(\zeta) \cdot (\nabla_{\theta_j} \log Pr(\zeta))\ d\zeta \\
&= \mathbb{E}[\mathcal{R}_j(\zeta) \cdot (\nabla_{\theta_j} \log Pr(\zeta))].
\end{aligned} \tag{4.26}$$

The probability $Pr(\zeta)$ can be expanded, that is,

$$Pr(\zeta) = \ p(s^1) \cdot \prod_{t=1}^{|T|} (\mathcal{P}(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1}) \cdot \prod_{j'=1}^{|S|} \pi_j'(a_j' \mid s^t; \theta_j')), \tag{4.27}$$

where $p(s^1)$ is the probability of the initial state s_1, $\prod_{t=1}^{|T|} (\mathcal{P}(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1}) \cdot \prod_{j'=1}^{|S|} \pi_j'(a_j' \mid s^t; \theta_j'))$ is the probability that the system has a sequence of states $s_1, s_2, ..., s_{|T|}$, and each agent $\mathcal{N}_j$ takes actions $a_j^1, a_j^2, ..., a_j^{|T|}$ at different time slots.

Pluging Eq. (4.27) into Eq. (4.26), we have

$$\begin{aligned}
\nabla_{\theta_j} J_j(\theta_j) &= \mathbb{E}[(\sum_{t=1}^{T} \mathcal{R}_j(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1})) \\
&\quad \cdot (\nabla_{\theta_j} \log\ p(s^1) \cdot \prod_{t=1}^{|T|} (\mathcal{P}(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1}) \cdot \prod_{j'=1}^{|S|} \pi_j'(a_j' \mid s^t; \theta_j')))] \\
&= \mathbb{E}[(\sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_j(a_j \mid s^t; \theta_j)) \cdot (\sum_{t=1}^{T} \mathcal{R}_j(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1}))].
\end{aligned} \tag{4.28}$$

However, it is impossible to calculate the expectation Eq. (4.28) directly. Therefore, we instead sample historical service request traces to estimate $\nabla_{\theta_j} J_j(\theta_j)$. That is,

$$\nabla_{\theta_j} J_j(\theta_j) \approx (\sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_j(a_j \mid s^t; \theta_j)) \cdot (\sum_{t=1}^{T} \mathcal{R}_j(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1})). \tag{4.29}$$

Notice that action a_t will affect the reward received prior to time slot t. Due to causality [118], we use the following estimation instead

$$\begin{aligned}\nabla_{\theta_j} J_j(\theta_j) &\approx \sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_j(a_j \mid s^t; \theta_j) \cdot (\sum_{t'=t}^{T} \mathcal{R}_j(s^t, a_1^t, ..., a_{|S|}^t, s^{t+1})) \\ &= \sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_j(a_j \mid s^t; \theta_j) \cdot G_j(t), \end{aligned} \tag{4.30}$$

To further decrease the estimated variance, we replace $G_j(t)$ with $G_j(t) - \mathbb{E}[G_j(t) \mid s^t]$, and the details of the proof of the replacement can be found in [6, 118]. We then have

$$\nabla_{\theta_j} J_j(\theta_j) \approx \sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_{\theta_j}(a_j^t \mid s^t; \theta_j)(G_j(t) - \mathbb{E}[G_j(t) \mid s^t]), \tag{4.31}$$

as an estimation of the gradient $\nabla_{\theta_j} J_j(\theta_j)$, and we deploy a critic network for each agent, which takes s^t as input and outputs a value function $V_j^{\pi_j}(s^t; \omega_j)$, as an approximate of $\mathbb{E}[G_j(t) \mid s^t]$.

Similarly, each critic network has a random parameter ω_j to be trained, we will train both the actor and critic networks of each agent $\mathcal{N}_j$ simultaneously. With the progress of training, the gap between $V_j^{\pi_j}(s^t; \omega_j)$ and $\mathbb{E}[G_j(t) \mid s^t]$ becomes smaller, which makes the actor network approaches the optimal policy.

The training proceeds as follows. Let R' be a set of historical service request traces. We simulate the service placement and request assignment processes, using historical service requests in R'.

In the service placement stage at each time slot t, each agent $\mathcal{N}_j$ first inputs the system state s^t into its actor network, its output $\pi_j(a_j \mid s^t; \theta_j^0)$ and service instance placement are obtained accordingly.

In the request assignment stage at time slot t, `Procedure 1` is invoked to assign the arrived requests in $R'(t)$. The system state s^t, action a_j^t and reward $\mathcal{R}_j^t$ of each agent $\mathcal{N}_j$ at each time slot t are recorded. Then, the service placement stage at time slot $t+1$ proceeds. This procedure is repeated until the last time slot T reaches.

We then update the actor and critic networks of each agent $\mathcal{N}_j$. Recall that we aim to maximize $J_j^t(\theta_j)$ at each time slot t, and we recorded the system state s^t, action a_j^t and reward $\mathcal{R}_j^t$ of each agent $\mathcal{N}_j$ at each time slot t.

To minimize the difference between the output $V_j^{\pi_j}(s^t; \omega_j)$ and $\mathbb{E}[G(t)|s^t]$, we first train the critic network by updating ω_j, followed by updating θ_j, i.e.,

$$\omega_j \leftarrow \omega_j - \eta \nabla_{\omega_j} (G_j(t) - V_j^{\pi_j}(s^t; \omega_j))^2, \tag{4.32}$$

and

$$\theta_j \leftarrow \theta_j + \eta \sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_{\theta_j}(a_j^t \mid s^t; \theta_j)(G_j(t) - V_j^{\pi_j}(s^t; \omega_j)). \tag{4.33}$$

Having updated both θ_j and ω_j, if they still do not converge, we can select another set R' of historical requests from the collection Γ randomly, and repeat the above steps on R' to further train the actor network and critic network of each agent $\mathcal{N}_j$ until the convergence of both θ_j and ω_j. The detailed process is shown in `Procedure` 3.

Procedure 3 Training procedure for the Actor network

Input: An MEC network $\mathcal{G} = (\mathcal{V} \cup S; E)$ with a set of cloudlets C, and a set of services F provided by the network and historical requests in the monitoring area.

Output: A set of trained Actor networks for all agents

for $j \leftarrow 1, 2, \ldots, |S|$ **do**
  Randomly initialize an actor network with parameter θ_j and a critic network with parameter ω_j for each agent;
while θ_j and ω_j has not converged **do**
  Select a set R' of service requests from request history;
  for $t \leftarrow 1, 2, \ldots, T$ **do**
    for $j \leftarrow 1, 2, \ldots, |S|$ **do**
      Calculate $\delta^t_{j,k} \quad \forall k \in \{1, 2, \ldots, |F|\}$ by calling `Procedure` 2;
    In request admission stage, calculate $x^{new}_{t,i,j}$ and $x^{idle}_{t,i,j}$ of requests in $R'(t)$ by invoking `Procedure` 1, and update the system according to the assignment.
    Calculate the rewards G^t_j for each agent;
  for $j \leftarrow 1, 2, \ldots, |S|$ **do**
    for $t \leftarrow 1, 2, \ldots, T$ **do**
      Input system state s^t into the critic networks;
      $\omega_j \leftarrow \omega_j - \eta \nabla_{\omega_j} (G_j(t) - V_j^{\pi_j}(s^t; \omega_j))^2$;
    $\theta_j \leftarrow \theta_j + \eta \sum_{t=1}^{T} \nabla_{\theta_j} \log \pi_{\theta_j}(a^t_j \mid s^t; \theta_j)(G_j(t) - V_j^{\pi_j}(s^t; \omega_j))$;
return The trained actor networks with parameters $\theta_1, \theta_2, ..., \theta_{|S|}$.

Having trained the DNNs, the actor networks then can be used for service placement. As shown in Fig. 4.3, at the service placement stage of each time slot t, we apply `Procedure` 2, where the agents take actions, i.e., decide the service placement on cloudlets based on the state of the MEC network. In Fig. 4.3, cloudlet 1 removes service instance 1, cloudlet 2 installs service instances 1 and 2, and cloudlet 3 installs service instance 2. At the request admission stage, we apply `Procedure` 1 to assign requests to appropriate cloudlets. As shown in Fig. 4.3, the request arrived at AP_4 is assigned to cloudlet 2, and the request that arrives at AP_5 is offloaded to cloudlet 3. We finally update the system state and proceed to the next time slot. The detailed algorithm is shown in `Algorithm` 11.

4.4 Performance evaluation

In this section, we evaluate the performance of the proposed algorithm for the online service placement and request assignment problem. We also analyze the im-

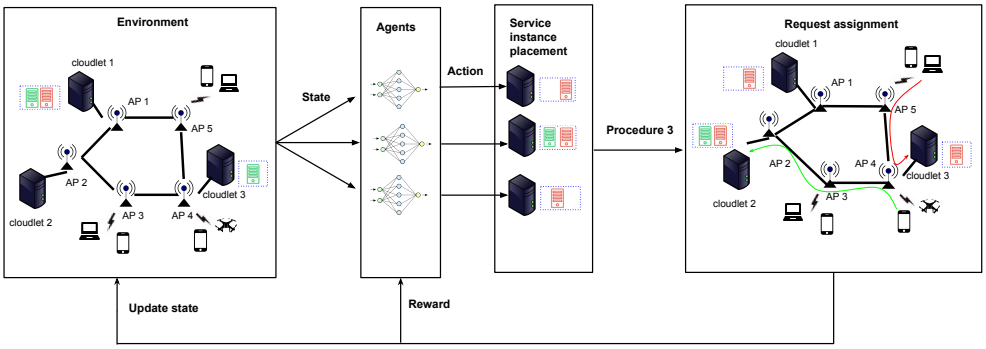


Figure 4.3: An illustration of the service placement and request assignment procedure at time slot t.

Algorithm 11 Advantage Actor-Critic based algorithm for online service placement and request assignment problem

Input: An MEC network $\mathcal{G} = (\mathcal{V} \cup \mathcal{S}; E)$, and a set of services F provided by the network, a set of requests R, and historical requests in the monitoring area.

Output: The service placement decision and request assignment at each time slot.

1: Train the actor networks by applying `Procedure` 3;
2: **for** $t \leftarrow 1, 2, \ldots, T$ **do**
3: **for** $j \leftarrow 1, 2, \ldots, |S|$ **do**
4: In service placement stage, call `Procedure` 2 to calculate $\delta^t_{j,k}$, $\delta^t_{j,k}$ instances for service f_k will be installed or deleted at Cloudlet c_j;
5: In request admission stage, calculate $x^{new}_{t,i,j}$ and $x^{idle}_{t,i,j}$ of requests in $R(t)$ by invoking `Procedure` 1, and update the system according to the assignment;
6: **return** The service placement decision $\delta^t_{j,k}$ derived from the actions a^t_j and the request assignments $x^{new}_{t,i,j}$, $x^{idle}_{t,i,j}$.

pact of important parameters on the performance of the proposed algorithm.

4.4.1 Experimental settings

We consider an MEC network with 100 APs, where 20% of them are equipped with cloudlets. The topologies of the network are randomly generated through a tool GT-ITM [62]. The computing capacity of each cloudlet is value randomly selected from 2,000 MHz to 4,000 MHz [52]. We assume that the MEC network provides 20 different types of services, and each different type of service instance requires 40 to 400 MHz computing resources randomly [52]. The network delay on the link between two APs is set a random value from 2 ms to 5 ms [87]. The instantiation delay of a service instance varies from 20 ms to 40 ms [92]. The instantiation cost of a service instance is randomly drawn within $[20, 50]$ [142]. The profit of processing

a user request per time slot is set within $[0.2, 0.3]$ per MHz [142], and the idle cost for a type of service instance is set within $[0.02, 0.03]$ per MHz per time slot. The request arrivals of a certain type of services at an AP follow a Poisson distribution, where the mean of Poisson distribution λ is randomly chosen within $[0.05, 0.1]$. The delay requirements of requests vary from 10 *ms* to 50 *ms*, and each request lasts from 1 to 5 time slots [52]. The value in each figure is the mean of the results out of 50 network instances of the same size, and the running time is obtained based on a machine with a 3.79GHz AMD Ryzen 5 CPU, RTX 2070 GPU and 32GB RAM. Unless otherwise specified, these parameters will be adopted in the default setting. Table 4.1 summarizes the default settings of the parameters in this chapter.

Table 4.1: Table of Parameter Settings

Parameters	Values
Number of APs	100
Number of cloudlets	20
Computing capacity	2,000 MHz - 4,000 MHz
Number of services	20
Required computing resource	40 MHz - 400 MHz
Network delay	2 *ms* - 5 *ms*
Instantiation delay	20 *ms* - 40 *ms*
Instantiation cost per MHz	20 - 50
Idle cost per MHz	20 - 50
Mean of arriving rate of requests per APs	0.05 - 0.1
Delay requirements	10 *ms* - 50 *ms*
Request duration	1 - 5 time slots

We evaluated `Algorithm` 11 against algorithm `Auto-regre` proposed in [142], where they utilized an auto-regression method to predict the number of required service instances as $n_{j,k}^t = 0.4 * n_{j,k}^{t-1} + 0.3 * n_{j,k}^{t-2} + 0.2 * n_{j,k}^{t-3} + 0.1 * n_{j,k}^{t-4}$. A baseline algorithm for service placement is also proposed, which releases idle service instances immediately when the demanded service by a request finishes, and it does not pre-install any service instances. We refer to this baseline algorithm as `NoCache`. For `Auto-regression` and `NoCache`, we propose a baseline algorithm for request assignment too, which assigns requests to cloudlets in descending order of their computing resource demands. For a given request r_i, we choose a set of cloudlets with sufficient computing capacities while meeting its service delay requirement. If there is not such a cloudlet, r_i will be rejected; otherwise, request r_i is assigned to the cloudlet with the maximum residual capacity and an idle instance for service f_{k_i}, i.e., r_i is assigned to a cloudlet c_j with the maximum cap_j by instantiating a new service instance on it. We also relax the ILP formulation to a linear programming (LP), and make use of the solution of the LP as an upper bound on the problem of concern. Even so, with the increase on network size (numbers of APs) or the number

of requests, the running time of algorithm LP dramatically grows, the LP algorithm thus is only applicable when the problem size is small or moderate. We refer to the LP solution as `LP`.

4.4.2 Performance evaluation of the online algorithm

We studied the performance of `Algorithm` 11 against algorithms `Auto-regression` and `NoCache`, respectively, by varying network size from 50 to 250. Fig. 4.4(a) plots the profits achieved by the three comparison algorithms and the upper bound on the optimal solution. It can be observed that `Algorithm` 11 delivers a solution which is from 76.4% to 80.5% of the upper bound of the optimal solution when network size varies from 50 to 100. Also, the profit by `Algorithm` 11 outperforms those delivered by algorithms `Auto-regression` and `NoCache` in all cases, and the performance gap between `Algorithm` 11 and the other comparison algorithms becomes larger with the increase on network size. Particularly, when there are 250 APs, `Algorithm` 11 receives 35.4% more profits than that of algorithm `Auto-regression`. The rationale behind is that `Algorithm` 11 has an advantage in the placement of service instances and delivers a better assignment of requests. Fig. 4.4(b) demonstrates the number of admitted requests by all algorithms. It can be seen that `Algorithm` 11 admits 24.5% more requests than that by algorithm `Auto-regression`, which shows that `Algorithm` 11 achieves better prediction of arriving requests.

Fig. 4.4(c) depicts the running times of the mentioned algorithms. Note that in reality, if the pattern of requests in the monitoring period does not dramatically change, we only need to conduct the training procedure once, as the trained DNNs can be re-used. Therefore, we exclude the training time from the running time of `Algorithm` 11. It can be seen that `Algorithm` 1 takes longer compared with algorithms `Auto-regression` and `NoCache`. Despite the running time of `Algorithm` 1 is high (637.12 seconds) when the network size is 250, the average running time per time slot is around 12.74 seconds, which is applicable since the proposed algorithm peoceeds in an online manner. Also, Procedure 2 can be applied before the arrivals of requests, which means the running time on determining service placement does not affect the latency of requests. Moreover, Procedure 2 can be run in cloudlets distributively, which further reduces the running time. Therefore, `Algorithm` 11 is applicable for latency-sensitive services and large-scale environments.

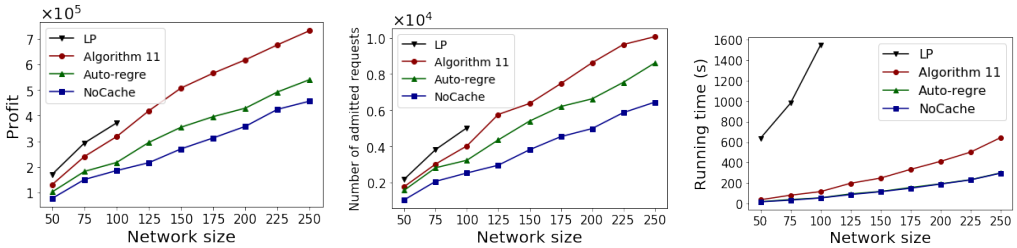


Figure 4.4: Performance of different algorithms by varying the size of the MEC network from 50 to 250.

4.4.3 Impact of important parameters on the performance of the online algorithm

We now investigate the impact of important parameters on algorithms: The arriving rate of requests and delay requirements of requests on the performance of the online algorithm `Algorithm` 11, by varying the mean of the Poisson distribution λ from $[0.025, 0.075]$ to $[0.1, 0.15]$. As shown in Fig. 4.5(a), `Algorithm` 11 always outperforms algorithms `Auto-regression` and `NoCache` under different arriving rates of requests. When the arriving rate of requests is set within $[0.1, 0.15]$, the profit of `Algorithm` 11 is 37.30% more than that of algorithm `NoCache`. It also can be seen that the profit gap between `Algorithm` 11 and algorithm `NoCache` becomes smaller when λ is drawn in the interval [0.0625, 0.1125]. The rationale behind this is that with the increase on arriving rates of requests, algorithm `NoCache` can accept more requests with flexible delay requirements, but `Algorithm` 11 cannot accept more requests due to the capacity constraint on each cloudlet. As shown in Fig. 4.5(b), `Algorithm` 11 admits the maximum number of user requests compared with the other algorithms. On the other hand, when the arriving rate of requests is in the interval $[0.075, 0.125]$, algorithm `NoCache` admits similar numbers of requests as algorithm `Auto-regression`, but algorithm `Auto-regression` collects less profits than those of the other two algorithms. The reason is that algorithm `Auto-regression` does not consider the profit of requests, so it pre-installs service instances with less profits. Instead, algorithm `NoCache` has more computing resources for admitting requests with high profits. Meanwhile, `Algorithm` 11 learns to pre-install service instances with higher profits, which leads to a higher accumulative profit. In addition, `Algorithm` 11 finishes within an acceptable time in Fig. 4.5(c).

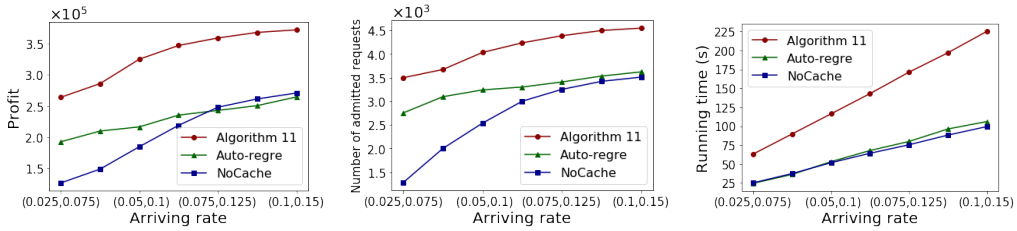


Figure 4.5: Performance of different algorithms by varying the Possion distribution parameter λ from the interval $[0.025, 0.075]$ to the interval $[0.1, 0.15]$.

We finally study the impact of delay requirements of requests on the performance of different algorithms, by varying the delay from $[10, 30]$ *ms* to $[10, 70]$ *ms*. Fig. 4.6(a) shows the performance of different algorithms in different delay requirement settings. We can see that `Algorithm` 11 delivers a solution with the maximum profit among the comparison algorithms. With more stringent delay requirements imposed, all the three comparison algorithms will deliver less profits. However, the profit reduction of `Algorithm` 11 is slower, while algorithm `NOCache` is fast. The reason is that `Algorithm` 11 pre-installs some instances, which shortens the service latency dramatically, and therefore it shows more tolerance to stringent delay requirements. From Fig. 4.6(b), we can also see that the number of admitted requests

increase with a longer maximum delay requirement. Fig. 4.6(c) depicts that the running times of the three comparison algorithms.

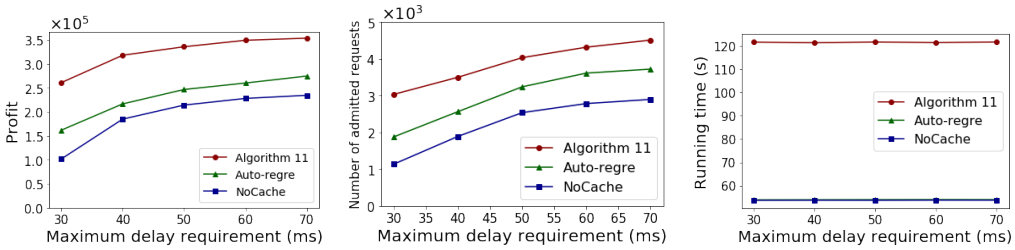


Figure 4.6: Performance of different algorithms by varying the maximum delay requirement from 30ms to 70ms.

4.5 Conclusions

In this chapter, we studied the problem of online service placement and request assignment in an MEC network while meeting user service delay requirements. We proposed a deep reinforcement learning-based mechanism to predict the most demanded services and their placements in a foreseeable future. We also devised an efficient online algorithm for dynamically assigning service requests to cloudlets with and without existing service instances. We finally conducted experiments through simulations. Experimental results demonstrate that the proposed algorithm is efficient and superior to the benchmark algorithms.

Chapter 5

Energy-Aware, Device-to-Device Assisted Federated Learning

In this chapter, we first formulate a energy-aware, D2D assisted federated learning problem with the aim to minimize. We then devise a near-optimal learning algorithm for the problem when the training data follows the i.i.d. data distribution. The crux of the proposed algorithm is to explore the energy usage of neighboring devices of each device for its local model uploading, by reducing the problem to a series of weighted maximum matching problems in corresponding auxiliary graphs. We also consider the problem without the i.i.d. data distribution assumption, for which we propose an efficient heuristic algorithm.

5.1 Introduction

To alleviate the communication overhead on uploading local models from devices to the edge server, in this chapter we introduce a device-to-device (D2D) assisted uploading concept to address this issue. That is, devices in the edge environments are paired according to their distances and energy availability. For each pair of devices, the one with less energy budget sends its trained local model to the one with more energy budget, and the received device then aggregates the model with its local model and uploads the aggregated model to the edge server. Since the wireless bandwidth capacity at the edge server is limited too, such a model aggregation and uploading can reduce the volume of data transmitted to the edge server. Performing energy-aware federated learning in edge environments thus poses several challenges: First, by allocating more energy on computation, a device can train its local model on a large volume portion of its collected data set, but this leaves the device less energy on its local model uploading or vice versa, how to strive for a non-trivial trade off of energy allocations between its local model training and local model transmission/uploading? Second, due to the heterogeneity of computing power and volume of collected data, different devices have different amounts of energy budgets at different rounds, the selection of device pairings heavily impacts not only the transmission energy consumption of devices but also the amount of data for local model training, thereby affecting the accuracy and convergence of federated

learning, how to pair devices such that the energy consumption is minimized while the training converge can be guaranteed is challenging. Finally, the wireless bandwidth capacity of the edge server usually is bounded, which can support only up to K devices rather than all devices to upload their local models to the server simultaneously if there are a large number of devices under the coverage of the server. As the contributions towards the global model of different devices are different, some devices are more important than others. Then, how to identify top-K contribution devices to upload their local models to the edge server? In the rest of this chapter we will tackle the aforementioned challenges.

The novelty of this chapter lies in that energy-aware device-to-device assisted federated learning in edge computing is considered through fully making use of an energy-sufficient neighbor device of each device for its local model uploading. A novel energy-aware, D2D-assisted federated learning problem is formulated, and a near-optimal algorithm for the problem is devised when the training data distribution meets certain assumptions.

The rest of the chapter is organized as follows. Section 5.2 introduces the system model, notions and notations, and defines the problem formally. Section 5.3 devises a near-optimal learning algorithm for a special case of the problem. Section 5.4 proposes an efficient algorithm for the problem without the training data distribution assumption. Section 5.5 evaluates the proposed algorithms empirically, and Section 5.6 concludes the chapter.

5.2 Preliminaries

In this section, we first introduce the system model. We then introduce federated learning in edge computing environments. We also introduce energy metrics on device assisted federated learning. We finally define the problem precisely.

5.2.1 System model

We consider a set of devices $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and an edge server s, assume that v_0 is the edge server s. Denote by $d_{i,j}$ the distance between devices v_i and v_j with $0 \leq i, j \leq |V|$. Assume that there is a DNN model deployed in the edge server s for training, and its training data comes from $|V|$ IoT devices. Each device $v_i \in V$ has a dataset $\mathcal{D}_i$ for the DNN model training. Denote by $\mathcal{D} = \cup_{i=1}^{|V|} \mathcal{D}_i$ the dataset of all devices. Each device v_i with energy capacity $\mathcal{E}_i$ has a set $\mathcal{P}$ of finite transmission power levels for communicating with other devices and the edge server, and such communication is conducted via D2D transmission, using one of its transmission power levels $p_i(t) \in \mathcal{P}$ at round t. We further assume that the federated learning process takes T training rounds, while at each round t, each device $v_i \in V$ performs its local model training for τ epochs, assuming that the energy budget $\mathcal{E}_i(t)$ ($\leq \mathcal{E}_i$) of v_i at round t is given [21], where $1 \leq t \leq T$. Due to limited wireless bandwidth on edge server s, we assume that at most K devices can upload their local models to the

server simultaneously at each round, where $1 \leq K \leq |V|$. An illustrative example of the system model is given in Fig. 5.1.

5.2.2 Federated learning in MEC

Let (x, y) be *one data point* in the dataset D, where $x \in \mathbb{R}^d$ is the input features and y is the label of the data point. We aim to train a deep neural network (DNN) to minimize the error between the output of the neural network and label y under a given input x. Specifically, the error is defined by a loss function $l(w, x, y)$, which is the mean squared error or cross-entropy[154], and $w \in \mathbb{R}^d$ is the *model parameter* of the DNN model that is a d-dimensional real vector.

The loss function on a dataset $\mathcal{D}$ is defined as follows.

$$L(w \mid \mathcal{D}) = \frac{\sum_{(x,y)\in\mathcal{D}} l(w, x, y)}{|\mathcal{D}|}. \tag{5.1}$$

The training objective is to find an optimal model parameter w^* to minimize the value of $L(w \mid \mathcal{D})$. Since it is difficult to find the closed-form solution of w^*, the training of neural networks often applies the gradient descent method to find a parameter w to approximate w^* as much as possible.

To protect the privacy of users, instead of training the DNN model by uploading the full dataset from all devices to the server, devices will train the deep neural network on their local datasets. At the beginning of each round t with $1 \leq t \leq T$, server s distributes the global model parameter $w(t-1)$ obtained at round $t-1$ to all devices, assuming that $w(0)$ is randomly initialized.

Due to limited wireless bandwidth capacity B of edge server s in the defined system model, it is impossible to allow all devices to upload their models to the server at the same time. Instead, a subset of devices $V_{fed}^t \subset V$ is chosen to participate in model training at each round t. Among the participating devices, up to K of the devices are allowed to upload their training models to the server directly without violating its bandwidth capacity constraint, where $K \geq 1$ is a positive integer. Specifically, each device $v_i \in V_{fed}^t$ uniformly samples a subset $\mathcal{S}_i(t)$ of its dataset $\mathcal{D}_i$ for local model training under its energy budget $\mathcal{E}_i(t)$ for round t, and then transmits its trained local model to its partner that is also in V_{fed}^t or vice versa. Its partner finally aggregates its local model with the received model and uploads the aggregated model to server s. Within each round t, we further assume that each device v_i applies τ gradient descent steps to train its local model $w_i(t)$, and we refer to each step of gradient descent of local training as *one epoch*. Denote by $w_i^k(t)$ the local model parameter of device v_i after the kth local training epoch with $k = 1, 2, \ldots, \tau$. Then, $w_i^0(t) = w(t-1)$ is the global model parameter distributed by server s after finishing local model training at round $t-1$. During each epoch k of round t with $1 \leq k \leq \tau$, device v_i updates its local model as follows.

$$w_i^k(t) = w_i^{k-1}(t) - \eta \cdot \nabla L_i(w_i^{k-1}(t) \mid \mathcal{S}_i(t)), \tag{5.2}$$

and

$$\nabla L_i(w_i^{k-1}(t) \mid \mathcal{S}_i(t)) = \frac{\sum_{(x,y)\in\mathcal{S}_i(t)} \nabla l(w_i^{k-1}(t), x, y)}{|\mathcal{S}_i(t)|}, \tag{5.3}$$

where η is the learning rate, and $\nabla l(w_i^{k-1}(t), x, y)$ is the gradient of the loss function $l(w, x, y)$ with respect to $w_i^{k-1}(t)$ on each data point $(x, y) \in \mathcal{S}_i(t)$ at epoch $(k-1)$ within round t.

Having τ-epoch local training at round t, some device $v_i \in V_{fed}^t$ uploads its trained (or aggregated) local model $w_i(t)$ $(= w_i^\tau(t))$ to server s. The uploaded local models from devices in V_{fed}^t are then aggregated at server s as follows [131].

$$w(t) = \frac{\sum_{v_i\in V_{fed}^t} |\mathcal{S}_i(t)| \cdot w_i^\tau(t)}{\sum_{v_i\in V_{fed}^t} |\mathcal{S}_i(t)|}. \tag{5.4}$$

Server s finally distributes the aggregated global model $w(t)$ back to each device in V at the beginning of the next round $t+1$.

5.2.3 Energy-budgeted D2D assisted uploading

Considering available energy heterogeneity on devices, some devices have less energy than others, due to more energy consumed on both uploading their local models to server s and their local model training. We here allow devices with sufficient energy to serve as relay nodes to help those devices with less energy to upload their local models to the server, thereby reducing the energy consumption of those less-energy devices. Meanwhile, the relay devices can aggregate the local models of relayed neighbors locally prior to uploading their aggregated local models to the server.

Specifically, each device $v_i \in V_{fed}^t$ has a destination device $\phi_{v_i}(t)$ at round t, which is either another device or server s. To avoid a long training delay, each device can only serve as either a relay or relayed node at each round exclusively. The transmission range $\theta_i(p_i(t))$ of device $v_i \in V$ at round t usually is determined by its transmission power level $p_i(t) \in \mathcal{P}$, a device v_j or server s can be the destination of device v_i only if it is within the transmission range of v_i, i.e., $d_{i,\phi_{v_i}(t)} \leq \theta_i(p_i(t))$. Denote by $\mathcal{C}_{v_i}(t)$ the set of nodes using v_i as their relays at training round t, we have $|\mathcal{C}_{v_i}(t)| \leq 1$. Similarly, $\mathcal{C}_s(t)$ is the set of nodes that can send their models to server s directly.

Since energy is the main constraint on devices, we assume that devices communicate with server s by adopting the Orthogonal Frequency-Division Multiplexing Access (OFDMA) mode. To avoid long delays on data transmission and aggregation and limited wireless bandwidth constraint on the edge server, we assume that at most K devices can send their local models to s with $1 \leq K \leq |V|$ at each round, that

is,

$$|\mathcal{C}_s(t)| \leq K. \tag{5.5}$$

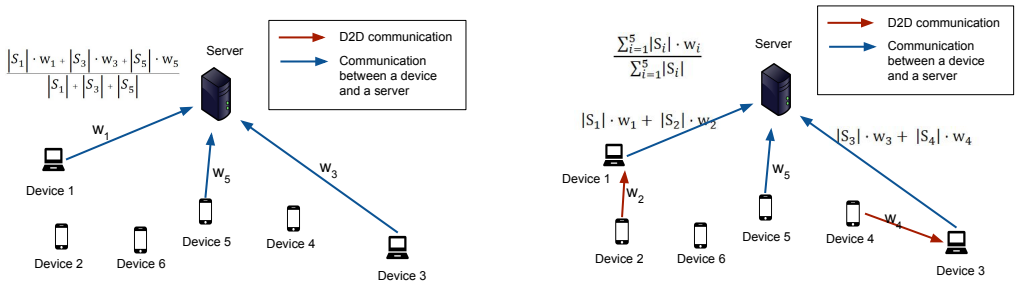
Having local training on its dataset $\mathcal{S}_i$ at round t, device $v_i \in V_{fed}^t$ then sends its local model $w_i^\tau(t)$ to its destination (a matching device of the device, which will be detailed later) $\phi_{v_i}(t)$ or server s. If its destination $\phi_{v_i}(t)$ is not to server s, device v_j $(= \phi_{v_i}(t))$ will aggregate its local model with its received local model from v_i if v_j, and transmits the aggregated result to server s to reduce transmission energy consumption. Denote by $w_i^g(t)$ the aggregated model uploaded by v_i. The aggregation at device v_i is

$$w_i^g(t) = |\mathcal{S}_i(t)| \cdot w_i^\tau(t) + \sum_{v_j \in \mathcal{C}_{v_i}(t)} |\mathcal{S}_j(t)| \cdot w_j^\tau(t),\ v_i \in V_{fed}^t \tag{5.6}$$

Recall that $\mathcal{S}_i(t)$ is a subset of dataset $\mathcal{D}_i$ of device v_i, $\mathcal{C}_{v_i}(t)$ is the set of devices that utilize v_i as their relay, and $\mathcal{C}_{v_i}(t)$ contains at most one device. If v_i participates in training at round t, $w_i^g(t)$ is the sum of sampled-data-size-weighted model parameters of v_i and the device in $\mathcal{C}_{v_i}(t)$ (if $\mathcal{C}_{v_i}(t)$ is not an empty set). The global model at server s after round t is updated as follows.

$$w(t) = \frac{\sum_{v_i \in \mathcal{C}_s(t)} w_i^g(t)}{\sum_{v_i \in V_{fed}^t} |\mathcal{S}_i(t)|}, \tag{5.7}$$

where the numerator of the right hand side of Eq. (5.7) is the weighted sum of model parameters of chosen devices in V_{fed}^t, while the denominator in the right hand side of Eq. (5.7) is the total number of sampled data points used in training. It can be seen that $w(t)$ in Eq. (5.7) is equivalent to it in Eq.(5.4).



(a) Traditional federated learning (b) D2D-assisted federated learning

Figure 5.1: Comparisons of the uploading phase between the traditional federated learning and the energy-aware D2D-assisted federated learning in an edge computing environment at one round with $K = 3$.

Fig. 5.1 provides a comparison of the uploading phase between the traditional federated learning and the energy-aware D2D-assisted federated learning, where at most three devices can communicate with a server at each round. For convenience,

in Fig. 5.1(b), the model training consists of two stages: the red arrows indicate the D2D communication in the first stage, while the blue arrows represent the model uploading communication between devices and the server in the second stage. The numbers on the arrows indicate the data volume of transmitted models. It can be seen from Fig. 5.1(a) that all (six) devices participate in local model training but only three of them can upload their trained local models to the edge server at each round, the trained results of the rest three devices are not uploaded, implying no contributions towards the global model. In contrast, it can be seen from Fig. 5.1(b) in the proposed solution that there are five devices participating in local model training and their trained local models are uploaded to the edge server through three of the five devices.

5.2.4 Energy consumption of devices

Let $h_{i,\phi_{v_i}(t)}$ be the channel gain between device v_i and device $\phi_{v_i}(t)$. The channel gain $h_{i,\phi_{v_i}(t)}$ [85] is

$$h_{i,\phi_{v_i}(t)} = \frac{\alpha}{d^2_{i,\phi_{v_i}(t)}}, \tag{5.8}$$

where α is the channel gain at the reference distance of 1 meter.

Denote by C the size of the DNN model w. By uploading its local model $w_i^\tau(t)$ to device $\phi_{v_i}(t)$ at round t, the amount of transmission energy consumed by device v_i is

$$Tran_{v_i}(t) = \frac{C \cdot p_i(t)}{B \cdot \log_2(1 + \frac{p_i(t) \cdot h_{i,\phi_{v_i}(t)}}{\sigma^2})}, \tag{5.9}$$

where σ is the white Gaussian noise power, and B is the wireless bandwidth capacity of server s.

Let ψ_i be the energy consumption of calculating the gradient on one data point at device v_i. As shown in Eq. (5.2) and Eq. (5.3), device v_i calculates the gradient of each data point in set $\mathcal{S}_i(t)$ for τ epochs at round t, the total computing energy consumption of v_i on local model training at round t is

$$Comp_{v_i}(t) = \psi_i \cdot |\mathcal{S}_i(t)| \cdot \tau. \tag{5.10}$$

The total energy consumption of device v_i at round t should be no greater than its energy budget $\mathcal{E}_i(t)$ for that round, i.e.,

$$Tran_{v_i}(t) + Comp_{v_i}(t) \leq \mathcal{E}_i(t). \tag{5.11}$$

5.2.5 Problem definition

Given a set of devices V and an edge server s, an integer K that is determined by the bandwidth of server s, they collaboratively perform federated learning to train a DNN model with a parameter vector (global model) w, each device $v_i \in V$ has a set $\mathcal{P}$ of transmission power levels. Each device v_i can sample a subset $\mathcal{S}_i(t)$ of its dataset $\mathcal{D}_i$ at each round t, and may upload its trained local model to server s directly or via another device, assuming that there are T rounds for the DNN model training. The *energy-aware D2D-assisted federated learning problem* in edge computing is to determine devices to participate in training at each round t, the number of sampled data points and the transmission power level of each chosen device, and the destination (paired) device of each chosen device to which to upload its local model at each round t with $1 \leq t \leq T$, such that the expected loss $\mathbb{E}[L(w(T) \mid \mathcal{D})]$ over the dataset $\mathcal{D}$ $(= \cup_{i=1}^{|V|} \mathcal{D}_i)$ is minimized, subject to the wireless bandwidth capacity B on server s, energy capacities and the maximum transmission ranges on devices.

The objective of the problem is to

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}[L(w(T) \mid \mathcal{D})], \\ \text{s.t.} \quad & (5.5)-(5.11), \end{aligned} \tag{5.12}$$

where $\mathbb{E}[L(w(T) \mid \mathcal{D})]$ is the expectation of $L(w \mid \mathcal{D})$ after T round iterations in Eq. (5.1).

5.3 Algorithm for a special federated learning problem

In this section, we consider a special case of the energy-aware D2D-assisted federated learning problem when $\tau = 1$ at each round t with $1 \leq t \leq T$. We devise a near-optimal learning algorithm for the problem, provided that the data set $\mathcal{D}_i$ of each device $v_i \in V$ follows i.i.d. data distribution. We also analyze the convergence of the proposed algorithm and the quality of the solution obtained.

5.3.1 Overview of the algorithm

Intuitively, if more data are used in the DNN model training, the trained DNN model can learn more accurate patterns on the dataset. We thus use as many sampled data points as possible to train the model with the aim to minimize the expected loss $\mathbb{E}[L(w(T) \mid D)]$ of the optimization objective in formula (5.12).

Following Eq.(5.10) and Eq.(5.11), when $\tau = 1$, the number $|\mathcal{S}_i(t)|$ of sampled data points that device v_i can use for its local model training at round t is upper bounded by

$$|\mathcal{S}_i(t)| \leq \frac{\mathcal{E}_i(t) - Tran_{v_i}(t)}{\psi_i}. \tag{5.13}$$

It can be seen from Inequality (5.13) that $|\mathcal{S}_i(t)|$ is bounded by the energy budget $\mathcal{E}_i(t)$ and the transmission energy consumption $Tran_{v_i}(t)$ of device v_i at round t. We then need to determine which devices to participate in model training at each round and the number of sample data points, the destination device, and the transmission power level of each chosen device.

According to Inequality (5.13), the decision at round t does not affect the number of sampled data points at any other round because the energy budget of each device for each round is given. Thus, the maximum number $|\cup_{v_i \in V_{fed}^t} \mathcal{S}_i(t)|$ of sampled data points used for local model training at each round t can be maximized, subject to energy budget $\mathcal{E}_i(t)$ for each $v_i \in V_{fed}^t$, where $1 \leq t \leq T$.

Since each device can help at most another device to relay its local model at each round, we can put devices in V into pairs so that the number of sampled data points for local model training at that round is maximized. To this end, we can reduce the problem to the maximum weight matching problem in an auxiliary graph. The detailed reduction is presented as follows.

5.3.2 Algorithm

From Eq. (5.9), the transmission energy consumption of a device increases with the increase in its transmission power level. To save energy of devices on transmissions, one device should select a minimum transmission power level in $\mathcal{P}$ such that its destination node is within the transmission range when it adopts the chosen transmission power.

For the sake of convenience, denote by $p_i^{min}(v_j)$ and $p_i^{min}(s)$ the minimum transmission power levels of v_i that v_j or s are within the transmission range of v_i. As device v_i has only limited power levels, $p_i^{min}(v_j)$ and $p_i^{min}(s)$ can be identified by binary search. Denote by $tran(v_i, v_j)$ and $tran(v_i, s)$ the amounts of energy consumed by transmitting the local model of v_i to device v_j or server s by adopting its minimum transmission powers, respectively, then

$$tran(v_i, v_j) = \frac{C \cdot p_i^{min}(v_j)}{B \cdot \log_2(1 + \frac{p_i^{min}(v_j) \cdot h_{i,v_j}}{\sigma^2})} \tag{5.14}$$

and

$$tran(v_i, s) = \frac{C \cdot p_i^{min}(s)}{B \cdot \log_2(1 + \frac{p_i^{min}(s) \cdot h_{i,s}}{\sigma^2})}, \tag{5.15}$$

where C is the size of the model w, and B is the bandwidth capacity of server s.

The proposed algorithm proceeds as follows. For each round t, a weighted auxiliary graph $G(t) = (U, E; w(\cdot, \cdot))$ is constructed, where $U = \{u_i, u_i' \mid v_i \in V\} \cup \{u_j^v \mid 1 \leq j \leq 2|V| - 2K\}$. The edge set E of $G(t)$ is defined as follows.

(i) For each device $v_i \in V$, if $\mathcal{E}_i(t) > tran(v_i, s)$, an edge $e(u_i, u_i')$ between u_i and u_i' is added to E, and its weight $w(u_i, u_i')$ is the maximum number of sampled data

points transferred if v_i directly sends its local model to server s, where $w(u_i, u_i') = \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,s)}{\psi_i} \rfloor$. Recall that the minimum transmission power of v_i is $tran(v_i, s)$ if v_i sends its local model to s, and the maximum amount of energy consumed for local model training is $\mathcal{E}_i(t) - tran(v_i, s)$.

(ii) For each pair of nodes $v_i \in V$ and $v_j \in V$ with $i \neq j$, denote by $S_{i,j}$ the maximum number of sampled data points from both devices and v_i is the relay node of v_j, we aim to find the maximum number of sampled data points if they both participate in model training at round t and one of them makes use of another as its relay vertex, then, $S_{i,j}$ is defined as follows.

$$S_{i,j} = \lfloor \frac{\mathcal{E}_i(t) - tran(v_i, v_j)}{\psi_i} \rfloor + \lfloor \frac{\mathcal{E}_j(t) - tran(v_j, s)}{\psi_j} \rfloor, \tag{5.16}$$

assuming that v_i is the relay device of v_j; otherwise, $S_{j,i}$ can be calculated similarly. If $max\{S_{i,j}, S_{j,i}\} > 0$, we add an edge (u_i, u_j) between u_i and u_j with weight $w(u_i, u_j) = max\{S_{i,j}, S_{j,i}\}$ to E, i.e., the maximum number of sampled data points of devices v_i and v_j is $w(u_i, u_j)$ when v_i is the relay node of v_j or vice versa.

(iii) For each pair of vertex u_i and dummy vertex u_j^ν, an edge $e(u_i, u_j^\nu)$ with an infinite weight $w(u_i, u_j^\nu)$ is added to E.

The edge set E of $G(t)$ can be partitioned into two disjoint subsets E_{dum} and E_{dev}, where E_{dum} consists of edges incident to at least one dummy vertex, i.e., $E_{dum} = \{e(u_i^\nu, u_j^\nu) \mid i \neq j,\ 1 \leq i, j \leq 2|V| - 2K\} \cup \{(u_i, u_j^\nu) \mid v_i \in V, 1 \leq j \leq 2|V| - 2K\}$, while E_{dev} contains only edges by device vertices, i.e., $E_{dev} = \{e(u_i, u_j) \mid v_i, v_j \in V, i \neq j\}$.

The construction of an auxiliary graph $G(t)$ is to find a set M of edges in $G(t)$ such that (i) M is a subset of E_{dev}; (ii) the cardinality $|M|$ of M is no larger than K; and (iii) M is a matching in $G(t)$ and the weighted sum of the edges in M is maximized. A set V_{fed}^t then can be derived from M, which is the set of endpoints of edges in M, i.e., $|V_{fed}^t| \leq 2K$.

Denote by $\mathcal{M}$ a weighted maximum matching in $G(t)$. Assume that M consists of only edges in $\mathcal{M}$ but not in E_{dum}, that is, $M = \mathcal{M} \backslash (E_{dum} \cap \mathcal{M})$. It can be seen that for device $v_i \in V$, the weight $w(u_i, u_i')$ of edge $e(u_i, u_i') \in \mathcal{M}$ represents the maximum number of sampled data points for its local model training if device v_i sends its local model to server s directly. For a pair of devices v_i and v_j, the weight $w(u_i, u_j)$ of edge $e(u_i, u_j) \in \mathcal{M}$ is the maximum number of sampled data points if both devices participate in the training at round t and one of them uses the other as its relay node to upload their aggregated local model to s.

Since $\mathcal{M}$ is a maximum weight matching in $G(t)$ that contains $4|V| - 2K$ vertices in total, there must be a matching edge in $\mathcal{M}$ incident to each of the $2|V| - 2K$ dummy vertices. Thus, there are $4|V| - 4K$ vertices in $\mathcal{M}$ with one endpoint being a dummy vertex.

Let V' be the set of the rest vertices in $G(t)$, clearly, $|V'| = 2K$. Let M' be the weighted maximum matching in the subgraph $G'(t) = (\{u_i, u_i' \mid v_i \in V'\}, E \cap \{(u_i, u_i') \cup (u_i, u_j)\} \mid v_i \in V', v_j \in V'\})$ of $G(t)$ induced by vertices in V'. Then, the cardinality of M' is no greater than K, i.e., $|M'| \leq K$, it can be seen that $M' \cup (\mathcal{M} \setminus M)$

is another matching of $G(t)$. As the weighted sum of edges in $\mathcal{M}$ is the maximum one, which equals the maximum number of sampled data points used for model training at round t, the weighted sum of edges in $M' \cup (\mathcal{M} \setminus M)$ must equal the weighted sum of edges in $\mathcal{M}$; otherwise, $\mathcal{M}$ is not the weighted maximum matching of $G(t)$. Thus, the set V_{fed}^t can be obtained from M', i.e., $|V_{fed}^t| \leq 2K$ as $|M'| \leq K$.

Let $\mathcal{M}$ be a weighted maximum matching in the auxiliary graph $G(t)$, by applying the weighted maximum matching algorithm in [51]. Let $V_{fed}^t = \emptyset$ initially, its construction is given as follows. For each edge $e(a, b) \in \mathcal{M}$, (i) if $a = u_i$ and $b = u_i'$, add device v_i to V_{fed}^t and set server s as the destination $\phi_{v_i}(t)$ of v_i; (ii) if $a = u_i$ and $b = u_j$ with $i \neq j$, both devices v_i and v_j are added to V_{fed}^t. Furthermore, if $S_{i,j} > S_{j,i}$, device v_j is the destination of v_i, v_i trains its local model and transmits its local model to v_j, v_j finally aggregates its local model and v_i's local model and uploads the aggregated model to server s; otherwise, device v_i is the destination of v_j, the similar operations can be performed, omitted; and (iii) if $a = u_i$ and $b = u_j^v$, then device v_i will not participate in training at round t.

Finally, each device $v_i \in V_{fed}^t$ sets its power level at $p_i^{min}(\phi_{v_i}(t))$, and samples $\lfloor \frac{\mathcal{E}_i(t) - Tran_{v_i}(t)}{\psi_i} \rfloor$ data points in set $\mathcal{D}_i$ for training at round t. The detailed algorithm for the special energy-aware D2D-assisted federated learning problem is shown in `Algorithm` 12.

We here use an example in Fig. 5.2 to illustrate how the proposed algorithm works at round t, where there are four devices ($|V| = 4$), and server s only allows two devices ($K = 2$) to upload its local model at each round. There are 4 (=$2|V| - 2K$) dummy vertices. As shown in Fig. 5.2, edges (u_1, u_2) and (u_2, u_3) represent device v_1 and v_3 can pair with v_2 to upload the trained local model, and edges (u_1, u_1'), (u_2, u_2') and (u_4, u_4') represent that device v_1, v_2 and v_4 can send their model to s alone. The red edges in Fig. 5.2 represent the edges in the weighted maximum matching $\mathcal{M}$. In Fig. 5.2, we assume that the weighted maximum matching of $G(t)$ consists of edges $e(u_1, u_2), e(u_4, u_4'), e(u_1', u_2^v), e(u_2', u_2^v), e(u_3', u_3^v)$,
and $e(u_3, u_4^v)$ respectively. Edge $e(u_4, u_4')$ is in the weighted maximum matching means that device v_4 participates in training at round t, and the number of sampled data points used for its local model training is determined by the weight of $e(u_1, u_1')$. Similarly, edge $e(u_1, u_2)$ is in the weighted maximum matching represents that both devices v_1 and v_2 participate in training at round t, and v_1 uses v_2 as its relay or vice versa. Device v_3 does not participate in training at round t as vertex u_3 are connected to a dummy vertex through edge $e(u_3', u_4^v)$ in the weighted maximum matching.

5.3.3 Algorithm analysis

The rest is to analyze the convergence rate and time complexity of the proposed algorithm, `Algorithm` 12, and show that the solution delivered by the proposed algorithm is a near-optimal solution under the well adopted assumptions on the loss function $L(\cdot)$ [131].

Algorithm 12 Algorithm for the special energy-aware, D2D-assisted federated learning problem

Input: A set of devices V, a server s, a local dataset $\mathcal{D}_i$ for each device $v_i \in V$, a set of transmission power level $\mathcal{P}$ and a DNN model w to be trained at each round t with $1 \leq t \leq T$.

Output: The set V_{fed}^t of devices participating in the training, the offloading destination $\phi_{v_i}(t)$, the set of sampled data points $\mathcal{S}_i(t)$, and the transmission power level $p_i(t)$ of each device v_i at each round t.

for $t \leftarrow 1$ to T **do**
  Initialize $G(t) \leftarrow (U, E)$ where $U = \emptyset$ and $E = \emptyset$;
  for $i \leftarrow 1$ to $|V|$ **do**
    $U \leftarrow U \cup \{u_i, u_i'\}$;
    Calculate $p_i^{min}(s)$ and $tran(v_i, s)$
    if $\mathcal{E}_i(t) > tran(v_i, s)$ **then**
      $E \leftarrow E \cup \{e(u_i, u_i')\}$;
      $w(u_i, u_i') \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,s)}{\psi_i} \rfloor$;
  for $i \leftarrow 1$ to $|V|$ **do**
    for $j \leftarrow i+1$ to $|V|$ **do**
      Calculate $p_j^{min}(s)$, $tran(v_i, v_j)$, $tran(v_j, s)$, $p_i^{min}(s)$, $tran(v_j, v_i)$, and $tran(v_i, s)$;
      if $tran(v_i, v_j) < \mathcal{E}_i(t)$ and $tran(v_j, s) < \mathcal{E}_j(t)$ **then**
        $S_{i,j} \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,v_j)}{\psi_i} \rfloor + \lfloor \frac{\mathcal{E}_j(t) - tran(v_j,s)}{\psi_j} \rfloor$;
      else
        $S_{i,j} \leftarrow 0$;
      if $tran(v_j, v_i) < \mathcal{E}_j(t)$ and $tran(v_i, s) < \mathcal{E}_i(t)$ **then**
        $S_{j,i} \leftarrow \lfloor \frac{\mathcal{E}_j(t) - tran(v_j,v_i)}{\psi_j} \rfloor + \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,s)}{\psi_i} \rfloor$;
      else
        $S_{j,i} \leftarrow 0$;
      $E \leftarrow E \cup \{e(u_i, u_j)\}$;
      $w(u_i, u_j) \leftarrow max\{S_{i,j}, S_{j,i}\}$;
  $U_{dum} \leftarrow \{u_j^v \mid 1 \leq j \leq 2|V| - 2K\}$
  $U \leftarrow U \cup U_{dum}$ /* U_{dum} is the set of dummy nodes */;
  for each dummy node $u_j^v \in U_{dum}$ **do**
    for each node $u_i \in U \backslash U_{dum}$ **do**
      $E \leftarrow E \cup \{e(u_i, u_j^v)\}$;
      $w(u_i, u_j^v) \leftarrow \infty$;
  Find a weighted maximum matching $\mathcal{M}$ in $G(t)$, by applying the algorithm in [51];
  for $i \leftarrow 1$ to $|V|$ **do**
    if $e(u_i, u_i') \in \mathcal{M}$ **then**
      $\phi_{v_i}(t) \leftarrow s$;
      $V_{fed}^t \leftarrow V_{fed}^t \cup \{v_i\}$;
      $|\mathcal{S}_i(t)| \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,s)}{\psi_i} \rfloor$;
    if $\exists u_j \in U, e(u_i, u_j) \in \mathcal{M}$ and $S_{i,j} \geq S_{j,i}$ **then**
      $\phi_{v_j}(t) \leftarrow s$; $\phi_{v_i}(t) \leftarrow v_j$;
      $V_{fed}^t \leftarrow V_{fed}^t \cup \{v_i, v_j\}$;
      $|\mathcal{S}_i(t)| \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i,v_j)}{\psi_i} \rfloor$;
      $|\mathcal{S}_j(t)| \leftarrow \lfloor \frac{\mathcal{E}_j(t) - tran(v_j,s)}{\psi_j} \rfloor$;
  return V_{fed}^t, $\phi_{v_i}(t)$, $|\mathcal{S}_i(t)|$ for each device v_i at round t.

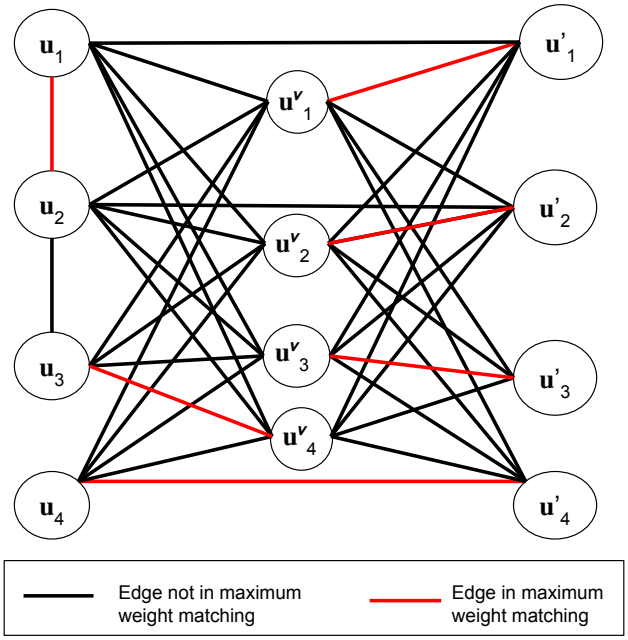


Figure 5.2: An example of auxiliary graph $G(t)$ when there are four devices and $K = 2$, two devices (either u_1 or u_2, and u_4) can upload their aggregated local models to server s at round t.

Assumption 1. The loss function $L(w)$ for the federated learning with model parameter w on a dataset $\mathcal{D}$ meets the following assumptions.

(1) The loss function $L(w)$ is convex.

(2) The loss function $L(w \mid \mathcal{D})$ is μ-smooth, that is, for any w_1 and w_2, we have $L(w_1 \mid \mathcal{D}) \leq L(w_2 \mid \mathcal{D}) + (w_1 - w_2) \cdot \nabla L(w_2|\mathcal{D}) + \frac{\mu}{2}||w_1 - w_2||^2$.

(3) The second derivative of $L(w)$ has a lower bound, that is, there exists a constant β such that $\nabla^2 L(w) \leq \beta \cdot I$, where I is an identity matrix.

(4) The learning rate η is smaller than β. i.e., $\eta < \beta$.

(5) $\tau = 1$.

(6) The datasets at the devices follow i.i.d distributions.

Notice that Assumption 1 from (1) to (4) holds when the DNN model is used for regression problems [131]. The later experimental result in Section 5.5 will show that the proposed algorithm is still applicable, even if the loss function $L(\cdot)$ does not abide by the mentioned assumptions in Assumption 1. Under Assumption 1, we have the following lemmas and theorems.

Lemma 3. The solution delivered by the proposed algorithm, `Algorithm` 12, is feasible. Let V_{fed}^t be the set of chosen devices at round t in the solution delivered by `Algorithm` 12, then set $|\mathcal{C}_s(t)|$ of devices uploading local model to s is no more than K as $|\mathcal{C}_s(t)| \leq K \leq |V_{fed}^t(t)| \leq 2K$, where $1 \leq t \leq T$.

Proof. It can be seen that the energy constraint on devices at each round has not been violated. The rest is to show that the number of devices uploading their local models to edge server s at each round is no larger than K.

For a given round t, an auxiliary graph $G(t) = (U, E; w(\cdot,\cdot))$ is constructed, in which there are $2|V|$ device vertices corresponding to the $|V|$ devices, and $2|V| - 2K$ dummy vertices. There is an edge between a dummy vertex and a device vertex and the edge is assigned with an infinite weight. Thus, each dummy vertex is adjacent to exactly one edge in the weighted maximum matching $\mathcal{M}$ of $G(t)$, which implies that there are $2|V| - 2K$ device vertices connected to the $2|V| - 2K$ dummy vertices by the edges in $\mathcal{M}$. Then, there are $2K$ device vertices in $G(t)$ that are not connected to any dummy vertices by the edges in $\mathcal{M}(t)$. We claim that these $2K$ device vertices form at most K edges in $\mathcal{M}$ as follows. Since edges in any matching of $G(t)$ do not share any common vertex, for each of these no more than K matching edges in $\mathcal{M}$, one of its corresponding two devices will upload its local model to server s directly, following the construction of $G(t)$. Thus, there are at most K devices communicating with server s at each round, i.e., $|\mathcal{C}_s(t)| \leq K$, where the participating device set V_{fed}^t is driven from the K device-to-device matching edges in $\mathcal{M}$. □

Lemma 4. In the solution delivered by `Algorithm` 12 for the special energy-aware D2D-assisted federated learning problem, the number of sampled data points from chosen devices for local model learning at each round t is maximized, where $1 \leq t \leq T$.

Proof. We show the claim by contradiction. Assume that there is another solution for the problem, in which the number of sampled data points is larger than the number of sampled data points in the solution delivered by `Algorithm` 12 at each round t. Let V_{fed}^{*t} be the set of devices participating in the training at round t, which is derived from the optimal solution with the maximum number of data points, and let $\phi_{v_i}^*(t)$, $p_i^*(t)$, and $\mathcal{S}_i^*(t)$ be the destination, the transmission power level, and the number of the sampled data points of device $v_i \in V_{fed}^t$.

We first show that the maximum number of sampled data points in $\cup_{v_i \in V_{fed}^{*t}} \mathcal{S}_i(t)$ equals the weighted sum of edges in a maximum weight matching of $G(t)$. For any device $v_i \in V^*{}_{fed}(t)$, if its destination $\phi_{v_i}^{*t}$ is server s, its sampled data volume $|\mathcal{S}_i^*(t)|$ must equal the weight of edge $e(u_i, u_i')$ as $w(u_i, u_i')$ is the maximum number of sampled data points of device v_i if v_i sends its local model to s directly; otherwise, if $|\mathcal{S}_i^*(t)| > w(u_i, u_i')$, we have $|\mathcal{S}_i^{*t}| \cdot \psi_i > \mathcal{E}_i(t) - Tran_{v_i}(t)$, which violates the energy constraint on v_i. Assume that $|\mathcal{S}_i^*(t)| < w(u_i, u_i')$, then device v_i can sample more data points than the current one without affecting other devices. This contradicts the assumption of the maximum number of sampled data points as the weight of edge (u_i, u_i'). Similarly, if v_i is the destination of device v_j with $i \neq j$, then $w(u_i, u_j) = |\mathcal{S}_i^*(t)| + |\mathcal{S}_j^*(t)|$. Therefore, there is a subset of edges in $G(t)$ such that the weighted sum of the edges in it is equal to the number of sampled data points $|\cup_{v_i \in V_{fed}^{*t}} \mathcal{S}_i^*(t)|$. Since a device can only serve as either the relay node of another device or the relayed node, not both of the roles at the same time. The set of edges through pairing relay

and relayed nodes forms a matching M^* of $G(t)$.

We then show that the weighted sum of edges in M^* is no larger than that of another matching $M' = \mathcal{M} \backslash E_{num}$ delivered by `Algorithm 12`. M^* consists of at most K edges due to constraint (5.5), which means $2K$ vertices in $\{u_i, u_i' \mid v_i \in V\}$ are adjacent to edges in M^*. Consequently, there are $2|V| - 2K$ vertices in $G(t)$ that are not adjacent to any edges in M^*. We can find a subset of edges $E^*_{dum} \subset E_{dum}$ with cardinality $2|V| - 2K$ such that $M^* \cup E^*_{dum}$ is still a matching in $G(t)$. Recall that $\mathcal{M}$ is a maximum weight matching in $G(t)$. Since the cardinality of the intersection of $\mathcal{M}$ and E_{num} is also $2|V| - 2K$ and the weights of edges in E_{num} are identical, the weighted sums of edges in $\mathcal{M} \cap E_{num}$ and E^*_{dum} are equal. As $M^* \cup E^*_{dum}$ is a matching, the weighted sum of edges in $M^* \cup E^*_{dum}$ is no larger than that of edges in $\mathcal{M}$. The weighted sum of edges in M^* thus is no larger than the weighted sum of edges in M'. The number of sampled data points in the solution delivered by `Algorithm 12` equals the weighted sum of edges in M^*. This contradicts the initial assumption that there is another solution that has more sampled data points than that of the solution delivered by `Algorithm 12`. □

Lemma 5. The expectation of loss $\mathbb{E}[L(w(t+1) \mid \mathcal{D})]$ is upper bounded by an additive value $\mathbb{E}[||w(t+1) - w^*||^2]$.

$$\mathbb{E}[L(w(t+1) \mid \mathcal{D})] \leq L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot \mathbb{E}[||w(t+1) - w^*||^2],$$

where w^* is the optimal model parameter and $L(w^* \mid \mathcal{D})$ is the minimum value of the loss function, which is constant.

Proof. From Assumption 1.(2), we have

$$\begin{aligned} L(w(t+1) \mid D) \leq & L(w^* \mid \mathcal{D}) + (w(t) - w^*) \cdot \nabla L(w^* \mid \mathcal{D}) + \frac{\mu}{2}||w(t+1) - w^*||^2 \\ = & L(w^* \mid \mathcal{D}) + \frac{\mu}{2}||w(t+1) - w^*||^2, \end{aligned}$$

where $\nabla L(w^* \mid \mathcal{D}) = 0$, since $L(w^* \mid \mathcal{D})$ is the minimum value and the gradient at a minimum point is 0. Therefore,

$$\begin{aligned} \mathbb{E}[L(w(t+1) \mid \mathcal{D})] \leq & \mathbb{E}[L(w^* \mid \mathcal{D}) + \frac{\mu}{2}||w(t+1) - w^*||^2] \\ = & L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot \mathbb{E}[||w(t+1) - w^*||^2], \end{aligned}$$

as the minimum loss $L(w^* \mid \mathcal{D})$ is a constant. □

Lemma 6. Let $w(t)$ be the result delivered by `Algorithm 12` at round t with $1 \leq t \leq T$. Then, the expected gap between $w(t)$ and the optimal one w^* is upper bounded, i.e.,

$$\mathbb{E}[||w(t+1) - w^*||^2] \leq (1 - \eta \cdot \beta) \cdot \mathbb{E}[||w(t) - w^*||^2] + \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}, \quad (5.17)$$

where Δ is the variance of $\nabla l(w(t), x, y)$.

Proof. As $\tau = 1$ at each round t, by `Algorithm` 12, we have

$$w_i^\tau(t) = w_i^1(t) = w(t-1) - \eta \cdot \nabla L_i(w(t-1) \mid \mathcal{S}_i(t)).$$

Recall that w^* is the optimal parameter, we have

$$\nabla L(w^* \mid \mathcal{D}) = 0.$$

Also, for any two models w_1 and w_2, there is a model w' such that

$$\nabla L(w_1 \mid \mathcal{D}) - \nabla L(w_2 \mid \mathcal{D}) = \nabla^2 L(w' \mid \mathcal{D}) \cdot (w_1 - w_2) \quad \forall w_1, w_2, \ \exists w',$$

due to the property of gradients. By Eq. 5.3,

$$\nabla L(w(t) \mid \cup_{v_i \in V_{fed}^t} \mathcal{S}_i(t)) = \frac{\sum_{v_i \in V_{fed}^t} |\mathcal{S}_i(t)| \cdot \nabla L_i(w(t) \mid \mathcal{S}_i(t))}{\sum_{v_i \in V_{fed}^t} |\mathcal{S}_i(t)|}$$

Therefore,

$$\begin{aligned}
w(t+1) - w^* &= \frac{\sum_{v_i \in V_{fed}^{t+1}} |\mathcal{S}_i(t+1)| \cdot w_i^\tau(t+1)}{\sum_{v_i \in V_{fed}^{t+1}} |\mathcal{S}_i(t+1)|} - w^* \\
&= \frac{\sum_{v_i \in V_{fed}^{t+1}} |\mathcal{S}_i(t+1)| \cdot (w(t) - \eta \cdot \nabla L_i(w(t) \mid \mathcal{S}_i(t+1)))}{\sum_{v_i \in V_{fed}^{t+1}} |\mathcal{S}_i(t+1)|} - w^* \\
&= w(t) - \eta \cdot \nabla L(w(t) \mid \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)) - w_i^* \\
&= w(t) - w^* - \\
&\quad \eta \cdot (\nabla L(w(t) \mid \mathcal{D}) - \nabla L(w^* \mid \mathcal{D}) - L(w(t) \mid \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)) - \nabla L(w(t) \mid \mathcal{D})), \\
&= w(t) - w^* - \\
&\quad \eta(\nabla^2 L(w' \mid \mathcal{D}) \cdot (w(t) - w^*)) - \eta(\nabla L(w(t) \mid \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)) - \nabla L(w(t) \mid \mathcal{D})).
\end{aligned}$$

Since we assume that the training data follows i.i.d, the gradient over the sampled data set can be regarded as the data set $\mathcal{D}$ [130]. We then apply central limit rules,

$$\frac{\nabla L(w(t) | \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1))) - \nabla L(w(t) \mid \mathcal{D})}{\Delta / \sqrt{| \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1))|}} \sim \mathcal{N}(0, 1),$$

where $\nabla l(w(t), x, y)$ is the gradient of the loss function $l(w(t), x, y)$ with respect to $w(t)$, and Δ^2 is the variance of $\nabla l(w(t), x, y)$.

The variance of $w(t+1) - w^*$ thus is

$$\begin{aligned}
&Var[w(t+1) - w^* \mid w(t)] = Var[w(t) - w^* - \eta(\nabla L(w(t) \mid \mathcal{D}) - \nabla L(w^* \mid \mathcal{D}))\\
&- \eta(\nabla L(w(t) \mid \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)) - \nabla L(w(t) \mid \mathcal{D}) \mid w(t)]\\
&= \eta^2 \cdot Var[(\nabla L(w(t) \mid \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)))\nabla L(w(t) \mid \mathcal{D}) \mid w(t)]\\
&= \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1))|},
\end{aligned}$$

We then have

$$\begin{aligned}
&||\mathbb{E}[w(t+1) - w^* \mid w(t)]||^2\\
&= ||\mathbb{E}[w(t) - w^* - \eta(\nabla^2 L(w' \mid \mathcal{D}) \cdot (w(t) - w^*)) \mid w(t)] +\\
&\mathbb{E}[-\eta(\nabla L(w(t)| \cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1) - \nabla L(w(t) \mid \mathcal{D})) \mid w(t)]||^2\\
&= ||\mathbb{E}[(I - \eta \cdot \nabla^2 L(w' \mid \mathcal{D})) \cdot (w(t) - w^*) \mid w(t)]||^2\\
&\leq ||\mathbb{E}[(1 - \eta \cdot \beta) \cdot (w(t) - w^*) \mid w(t)]||^2 \qquad (5.18)\\
&= (1 - \eta \cdot \beta) \cdot ||w(t) - w^*||^2
\end{aligned}$$

where Inequality (5.18) is due to Assumption 1.2. We then have

$$\begin{aligned}
&\mathbb{E}[||w(t+1) - w^*||^2 \mid w(t)]\\
&= ||\mathbb{E}[w(t+1) - w^* \mid w(t)]||^2 + Var[w_s(t+1) - w^* \mid w(t)]\\
&\leq (1 - \eta \cdot \beta) \cdot ||w(t) - w^*||^2 + \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}.
\end{aligned}$$

By the law of the total expectation,

$$\mathbb{E}[||w(t+1) - w^*||^2] \leq (1 - \eta \cdot \beta) \cdot \mathbb{E}[||w(t) - w^*||^2] + \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}.$$

□

Theorem 5.1. *Assuming that loss function $L(\cdot)$ is μ-smooth and β-Lipschitz. The proposed algorithm,* `Algorithm` *12, delivers a feasible solution for the special energy-aware D2D assisted federated learning problem with the expected loss $\mathbb{E}[L(w(T) \mid \mathcal{D})]$, which is defined as follows.*

$$\begin{aligned}
&\mathbb{E}[L(w(T) \mid \mathcal{D})]\\
&\leq L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot ((1 - \eta \cdot \beta)^T \cdot \mathbb{E}[||w(0) - w^*||^2]\\
&+ \sum_{t=0}^{T-1} (1 - \eta \cdot \beta)^{T-t-1} \cdot \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}), \qquad (5.19)
\end{aligned}$$

where μ, η, and β are given constants that are defined in Assumption 1, and Δ^2 is the variance of $\nabla l(w(t), x, y)$ for all data points $(x, y) \in \mathcal{D}$.

Proof. Having Lemma 5 and 6, Theorem 5.1 is shown as follows.

From Lemma 5, we can see that the expectation of loss $\mathbb{E}[L(w(t+1) \mid \mathcal{D})]$ depends on $\mathbb{E}[||w(t+1) - w^*||^2]$. This implies that minimizing the value $\mathbb{E}[||w(T) - w^*||^2]$ is equivalent to minimizing the expectation loss $\mathbb{E}[L(w(T) \mid \mathcal{D})]$ at the last round T.

By Lemma 6, the expectation $\mathbb{E}[||w(t) - w^*||^2]$ at round t is upper bounded by $\mathbb{E}[||w(t-1) - w^*||^2]$ and $\frac{\eta^2 \cdot \Delta'^2}{|\cup_{v_i \in V_{fed}^t} \mathcal{S}_i(t)|}$. It can be seen that the value gap between the global model parameter and the optimal model parameter is bounded by the difference of the global model parameter to the optimal model parameter at round $t-1$ and the number of sampled data points at round t. By expanding Inequality (5.17) recursively, we then have

$$\begin{aligned}
&\mathbb{E}[||w(t+1) - w^*||^2] \\
&\leq (1 - \eta \cdot \beta) \cdot \mathbb{E}[||w(t) - w^*||^2] + \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|} \\
&\leq (1 - \eta \cdot \beta)^{t+1} \cdot \mathbb{E}[||w(0) - w^*||^2] + \sum_{t'=0}^{t} \frac{\eta^2 \cdot \Delta^2 \cdot (1 - \eta \cdot \beta)^{t-t'}}{|\cup_{v_i \in V_{fed}^{t'+1}} \mathcal{S}_i(t'+1)|}.
\end{aligned} \tag{5.20}$$

By plugging (5.20) and the final round T into the optimization objective (5.12), we have

$$\begin{aligned}
&\mathbb{E}[L(w(T) \mid \mathcal{D})] \\
&\leq L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot ((1 - \eta \cdot \beta)^T \cdot \mathbb{E}[||w(0) - w^*||^2] + \sum_{t=0}^{T-1} \frac{\eta^2 \cdot \Delta^2 \cdot (1 - \eta \cdot \beta)^{T-t-1}}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}).
\end{aligned}$$

□

It can be seen from Theorem 5.1 that the accuracy of the obtained solution increases with the increase in the number of sampled data points $|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|$, while the latter is proportional to the value of K, where a larger K implies more devices can participate in the training at each round t, thereby leading to more sampled data points for model learning.

To better understand the relationship between K and the number of data sample points $|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|$ in the solution delivered by the proposed algorithm, `Algorithm` 12, we consider an extreme case of the special energy-aware D2D-assisted federated learning problem, where each device v_i has the same number of data points $|\mathcal{D}'|$, i.e., $|\mathcal{D}_i| = |\mathcal{D}'|$ for every device v_i. We further assume that server s is within the transmission range of all devices but none of the devices is within the transmission range of others. Under this assumption, each device can only upload its local model to server s directly. If the energy budget $\mathcal{E}_i(t)$ of each device v_i suffices for training

at most half the number of data points in $\mathcal{D}_i$ at round t, i.e., $|\mathcal{S}_i(t)| = \frac{\mathcal{D}'}{2}$. then the expected loss of the DNN model on dataset $\mathcal{D}$ is

$$\begin{aligned}
&\mathbb{E}[L(w(T) \mid \mathcal{D})] \\
&\leq L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot ((1-\eta\cdot\beta)^T \cdot \mathbb{E}[||w(0) - w^*||^2] \\
&+ \sum_{t=0}^{T-1} (1-\eta\cdot\beta)^{T-t-1} \cdot \frac{\eta^2 \cdot \Delta^2}{|\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|}) \\
&= L(w^* \mid \mathcal{D}) + \frac{\mu}{2} \cdot ((1-\eta\cdot\beta)^T \cdot \mathbb{E}[||w(0) - w^*||^2] \\
&+ \sum_{t=0}^{T-1} (1-\eta\cdot\beta)^{T-t-1} \cdot \frac{2\eta^2 \cdot \Delta^2}{K \cdot |\mathcal{D}'|}). \qquad (5.21)
\end{aligned}$$

It can be seen from Inequality (5.21) that a more accurate solution can be obtained when the value of K is larger, provided that the number T of training rounds does not change at all.

Theorem 5.2. *Given a set V of devices and an edge server s for a federated learning process with T round training, there is an algorithm,* `Algorithm` *12, for the special energy-aware D2D-assisted federated learning problem, which can deliver a near optimal solution within $\mathcal{O}(T \cdot |V|^4)$ time.*

Proof. By `Algorithm` 12, the number of sampled data points at each round t is maximized. While the number of sampled data points at one round is independent of other rounds, `Algorithm` 12 delivers a solution with maximizing the value of $\sum_{t=0}^{T-1} |\cup_{v_i \in V_{fed}^{t+1}} \mathcal{S}_i(t+1)|$, which is equivalent to minimizing Eq. (5.19).

We then analyze the time complexity of `Algorithm` 12. The construction of auxiliary graph $G(t)$ at each round t takes $\mathcal{O}(|V|^2)$ time, as $G(t)$ consists of $4|V| - 2K$ vertices. Finding a maximum weight matching in $G(t)$ takes $\mathcal{O}((4|V| - 2K)^4) = \mathcal{O}(|V|^4)$ time [51]. The time complexity of the proposed algorithm is $\mathcal{O}(T \cdot |V|^4)$, since it takes T training rounds. □

5.4 Algorithm for the D2D-assisted federated learning problem

So far we assumed that the data distribution of each device follows i.i.d., and the proposed algorithm for a special energy-aware, D2D-assisted federal learning problem, trains local models within one epoch at each round. In this section, we present an efficient heuristic algorithm for the energy-aware, D2D-assisted federated learning problem, by removing the mentioned assumptions on data sets and allowing multiple-epoch local training. We start with choosing devices for uploading their local models at each round, we then give an overview of the proposed algorithm, followed by detailing the algorithm.

5.4.1 Device choices at each round

In reality, the data generated by different devices are biased and may not follow the i.i.d. assumption. The global optimal model parameter w^* is different from the local optimal model parameter w_i^* with the minimum local loss, i.e., the local loss $L(w^* \mid \mathcal{D}_i)$ is not optimal in terms of the global model. Since each device v_i only samples data points in its dataset $\mathcal{D}_i$, device v_i can only train the model w towards w_i^*. Moreover, only up to K devices are chosen to upload their local models to server s at each round, which makes the global model w strongly biased towards these chosen devices. To mitigate this biased model training caused by some but all devices in V participating in training on the global model, we propose an efficient strategy for device choices as follows.

A device will have a large variance if it samples inadequate numbers of data points for local training. To ensure the quality of local training, it requires each chosen device to train on at least $\delta \cdot |\mathcal{D}_i|$ data points with $\delta \cdot |\mathcal{D}_i| \leq |\mathcal{S}_i(t)|$, where δ is a given threshold with $0 < \delta < 1$. Intuitively, a higher local loss on some devices implies that the model lacks of proper training on the data samples of the chosen devices. Cho *et al.* showed that devices with higher local losses make the learning convergence faster [25]. This implies that devices with higher local losses will have higher priorities to participate in training at that round.

We here adopt the similar principle as Cho *et al.* [25] did. The key is to choose a subset V_{fed}^t of devices with high local loss to participate in training at each round t. To this end, each device v_i samples a small portion of its dataset and makes use of the sampled data points to estimate the local loss $\hat{L}(w_i(t-1) \mid \mathcal{D}_i)$ at the beginning of the next round t. Note that the energy consumption of devices on inferences usually is much smaller than that of it on model training. We thus assume that the energy consumption of devices on inference estimation is negligible. A device v_i that does not participate in training at round t has either a lower local loss than those of devices in V_{fed}^t, or a higher local loss than some of devices in V_{fed}^t, but may cause that devices with higher local losses have inadequate energy to train. In the following we use an example to illustrate why the greedy-based service choice for set V_{fed}^t does not work.

Consider that there are three devices v_{i_1}, v_{i_2} and v_{i_3}, where v_{i_1} has a higher local loss than v_{i_2}, and v_{i_2} has a higher local loss than v_{i_3} but both v_{i_1} and v_{i_2} are far from server s (i.e., server s is not in their transmission ranges). We further assume that device v_{i_3} is the only device within the transmission ranges of both v_{i_1} and v_{i_2} while server s is within the transmission range of v_{i_3}. In this case, only either v_{i_1} or v_{i_2} can participate in training at a round, and v_{i_3} as the unique relay vertex of either of them can upload the local model to server s at round t. Therefore, devices v_{i_1} and v_{i_3} form a pair, and v_{i_3} is added to V_{fed}^t despite that v_{i_2} has a higher local loss than that of v_{i_3}, i.e., $V_{fed}^t = \{v_{i_1}, v_{i_3}\}$.

5.4.2 Algorithm overview

Due to the energy budget on each device at each round, it does not form a feasible solution by adding devices to set V_{fed}^t greedily. Instead, finding a feasible solution to the problem needs taking both the estimated local loss and the energy budget of each device into consideration, when determining whether the device can be added to V_{fed}^t.

The basic idea behind the proposed algorithm is similar to the one in the previous section, i.e., we reduce the problem to a series of maximum weight matching problems in different auxiliary graphs. Specifically, we start by sorting devices in V in non-decreasing order of their estimated local loss. For the sake of convenience, let $v_1, v_2, \ldots, v_{|V|}$ be the sorted devices, where v_1 and $v_{|V|}$ have the lowest and highest local losses, respectively.

We then construct an auxiliary graph $\mathcal{G}(t) = (U, E; w(\cdot, \cdot))$ that is similar to the auxiliary graph in the previous section at round t, where the vertex set U consists of vertices u_i and u_i' for each device $v_i \in V$ and $2|V| - 2K$ dummy vertices.

The construction of the edge set E of $\mathcal{G}(t)$ is as follows. For each device $v_i \in V$, to fulfill the number of sampled data points, v_i must spend at least the amount $\psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau$ of energy on training. For the sake of convenience, denote by $\mathcal{E}_i'(t)$ $(= \mathcal{E}_i(t) - \psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau)$ the energy budget of v_i at round t. The required transmission power $tran(v_i, s)$ of v_i then is calculated as follows.

If the required transmission energy constraint can be met (i.e., $tran(v_i, s) \leq \mathcal{E}_i(t) - \mathcal{E}_i'(t)$), add an edge $e(u_i, u_i')$ to E in $\mathcal{G}(t)$ with weight of $w(u_i, u_i')$ $(= 2^i)$, assuming that device v_i has the ith lowest estimated local loss. An edge $e(u_i, u_i')$ with weight 2^i indicates the high priority of v_i as a potential uploading device at round t, and v_i can upload its local model to s directly if it is chosen.

For each pair of devices v_i and v_j with $i > j$, if $tran(v_i, v_j) \leq \mathcal{E}_i(t) - \mathcal{E}_i'(t)$ and $tran(v_j, v_s) \leq \mathcal{E}_i(t) - \mathcal{E}_j'(t)$, add an edge $e(u_i, u_j)$ to E with weight of $w(u_i, u_j)$ $(= 2^i + 2^j)$. This means that v_i can send its trained local model to v_j, v_j then aggregates the received model from v_i with its own local model, and uploads the aggregated model to server s. Otherwise, if $tran(v_j, v_i) \leq \mathcal{E}_i(t) - \mathcal{E}_j'(t)$ and $tran(v_i, v_s) \leq \mathcal{E}_i(t) - \mathcal{E}_i'(t)$), we add an edge $e(u_i, u_j)$ to E with weight of $w(u_i, u_j)$ $(= 2^i + 2^j)$, the operations are similar, instead, v_i will upload the aggregated model to server s.

For a pair of a dummy vertex u_j^v and a device vertex u_i, add an edge $e(u_j^v, u_i)$ with weight of ∞ to E.

Let $\mathcal{M}(t)$ be the maximum weight matching in $\mathcal{G}(t)$, the K chosen devices driven from $\mathcal{M}(t)$ form a solution of the problem, and the set V_{fed}^t of devices for uploading their local models to server s can also be obtained, too.

5.4.3 Algorithm

The proposed algorithm proceeds as follows.

At the beginning of each round t, each device v_i samples a small subset of its dataset $\mathcal{D}_i$ to estimate the local loss $\hat{L}(w(t-1) \mid \mathcal{D}_i)$. A weighted auxiliary graph

$\mathcal{G}(t)$ then is constructed, and the estimated local loss and device ranking are used to assign weights to the edges in the graph. A weighted maximum matching $\mathcal{M}(t)$ in $\mathcal{G}(t)$ then is found, and a feasible solution to the problem finally is derived from the weighted maximum matching $\mathcal{M}(t)$. That is, if edge $e(u_i, u_i') \in \mathcal{M}(t)$, device v_i uploads its local model to server s directly; otherwise, if edge $e(u_i, u_j) \in \mathcal{M}(t)$ with $i > j$ and $tran(v_j, s) \leq \mathcal{E}_i(t) - \mathcal{E}_i'(t)$, device v_j is the destination of v_i, aggregates its local model with the local model of v_i and uploads the aggregated local model to s; Similarly, if $e(u_i, u_j) \in \mathcal{M}(t)$ with $i > j$ but $tran(v_j, s) > \mathcal{E}_i(t) - \mathcal{E}_i'(t)$, v_j sends its local model to v_i for aggregation and v_i uploads the aggregated model to server s. The rest devices that are not incident to any matching edge in $\mathcal{M}(t)$ will not participate in training at round t. As a result, the number of sampled data points $|\mathcal{S}_i(t)|$ of device v_i at round t is $\lfloor \frac{\mathcal{E}_i(t) - Tran_{v_i}(t)}{\psi_i} \rfloor$.

The detailed algorithm for the energy-aware, D2D assisted federated learning problem is given in `Algorithm` 13.

Theorem 5.3. *Given a set V of IoT devices, and an edge server s that performs federated learning training within T rounds, there is an efficient algorithm,* `Algorithm` *13, for the energy-aware D2D-assisted federated learning problem, which takes $\mathcal{O}(T \cdot |V|^4)$.*

Proof. The analysis of time complexity of `Algorithm` 13 is similar to `Algorithm` 12, while the latter has been analyzed in Theorem 5.2, omitted. □

5.5 Performance evaluation

In this section, we evaluated the performance of the proposed algorithms through experimental simulations. We also investigated the impact of important parameters on the performance of the proposed algorithms.

5.5.1 Experimental Settings

We consider a sensor network that consists of 100 devices randomly deployed in a circular area with a 250 meter radius and an edge server co-located with an access point at the center of the area, where the server can communicate with up to 20% of the devices directly, i.e. $K = |V| * 20\%$. The devices and the server collaboratively conduct a federated learning model training with 50 rounds, and each training round consists of $\tau = 1$ epoch. The bandwidth at each AP is set at 10 MHz and the white noise σ^2 is set at 1×10^{-10} Watt [61]. The channel gain at the reference distance of 1 meter α is set as 10^{-3} [85]. The maximum transmission power of IoT devices in [61] is 0.5 Watt, The transmission power levels of each device are in $\{0.2, 0.3, 0.4, 0.5\}$ Watt and the minimum signal-to-noise ratio is at 14dB [129]. The corresponding transmission distances of these transmission power levels are 282, 346, 400, and 447 meters, respectively. The energy budget on each device at each round is randomly drawn in [0.01, 0.04] Joules [21]. The dataset in this experiment is the MNIST that consists of 70,000 images of handwritten digits, which is a well known dataset adopted by

Algorithm 13 Algorithm for the energy-aware, D2D-assisted federated learning problem

Input: A set of devices V, a server s, the dataset $\mathcal{D}_i$ of each device $v_i \in V$, an edge budget $\mathcal{E}_i(t)$ of v_i at round t, a set of transmission power level $\mathcal{P}$, a given percentage of sampled data threshold δ and a DNN model w to be trained at t round with $1 \leq t \leq T$.

Output: The set V_{fed}^t of devices that participate in training, the offloading destination $\phi_{v_i}(t)$, the set of sampled data $\mathcal{S}_i(t)$, and the transmission power level $p_i(t)$ of each chosen device v_i at each round t.

for $t \leftarrow 1$ to T **do**
  Each device $v_i \in V$ selects a small subset of its data to estimate the local loss $\hat{L}(w(t-1) \mid \mathcal{D}_i)$;
  Sort vertices in V in non-decreasing order of the estimated local loss;
  Initialize $G(t) \leftarrow (U, E)$ where $U = \emptyset$ and $E = \emptyset$;
  for $i \leftarrow 1$ to $|V|$ **do**
    $U \leftarrow U \cup \{u_i, u_i'\}$;
    Calculate $p_i^{min}(s)$, $tran(v_i, s)$;
    if $tran(v_i, s) + \psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau \leq \mathcal{E}_i(t)$ **then**
      $E \leftarrow E \cup \{e(u_i, u_i')\}$; $w(u_i, u_i') \leftarrow 2^i$;
  for $i \leftarrow 1$ to $|V|$ **do**
    for $j \leftarrow i+1$ to $|V|$ **do**
      Calculate $p_j^{min}(s)$, $tran(v_i, v_j)$, and $tran(v_j, s)$;
      if $tran(v_i, v_j) + \psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau < \mathcal{E}_i(t)$ && $tran(v_j, s) + \psi_j \cdot \delta \cdot |\mathcal{D}_j| \cdot \tau < \mathcal{E}_j(t)$ || $tran(v_j, v_i) + \psi_j \cdot \delta \cdot |\mathcal{D}_j| \cdot \tau < \mathcal{E}_j(t)$ && $tran(v_i, s) + \psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau < \mathcal{E}_i(t)$ **then**
        $E \leftarrow E \cup \{e(u_i, u_j)\}$;
        $w(u_i, u_j) \leftarrow 2^i + 2^j$;
  $U_{dum} \leftarrow \{u_j^v \mid 1 \leq j \leq 2|V| - 2K\}$
  $U \leftarrow U \cup U_{dum}$ /* U_{dum} is the set of dummy nodes */;
  for each dummy vertex $u_j^v \in U_{dum}$ **do**
    for each vertex $u_i \in U \backslash U_{dum}$ **do**
      $E \leftarrow E \cup \{e(u_i, u_j^v)\}$; $w(u_i, u_j^v) \leftarrow \infty$;
  Find a weighted maximum matching $\mathcal{M}(t)$ in $G(t)$, by applying the algorithm in [51];
  for $i \leftarrow 1$ to $|V|$ **do**
    if $e(u_i, u_i') \in \mathcal{M}(t)$ **then**
      $\phi_{v_i}(t) \leftarrow s$; $V_{fed}^t \leftarrow V_{fed}^t \cup \{v_i\}$;
      $|\mathcal{S}_i(t)| \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i, s)}{\psi_i} \rfloor$;
    if $\exists j > i, s.t. e(u_i, u_j) \in \mathcal{M}(t)$ **then**
      $V_{fed}^t \leftarrow V_{fed}^t \cup \{v_i, v_j\}$;
      if $tran(v_i, s) + \psi_i \cdot \delta \cdot |\mathcal{D}_i| \cdot \tau < \mathcal{E}_i(t)$ **then**
        $\phi_{v_i}(t) \leftarrow s$; $\phi_{v_j}(t) \leftarrow v_i$;
        $|\mathcal{S}_j(t)| \leftarrow \lfloor \frac{\mathcal{E}_j(t) - tran(v_j, v_i)}{\psi_j} \rfloor$;
        $|\mathcal{S}_i(t)| \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i, s)}{\psi_i} \rfloor$;
      else
        $\phi_{v_j}(t) \leftarrow s$; $\phi_{v_i}(t) \leftarrow v_j$;
        $|\mathcal{S}_j(t)| \leftarrow \lfloor \frac{\mathcal{E}_j(t) - tran(v_j, s)}{\psi_j} \rfloor$;
        $|\mathcal{S}_i(t)| \leftarrow \lfloor \frac{\mathcal{E}_i(t) - tran(v_i, v_j)}{\psi_i} \rfloor$;
return V_{fed}^t, $\phi_{v_i}(t)$, $\mathcal{S}_i(t)$ for each device v_i at round t.

many federated learning works [21, 131]. We randomly distribute the training images of the dataset to 100 IoT devices. The DNN model used in this experiment is Le-net5, which has a size $\mathcal{C}$ of 1,960Kbits [57]. The energy consumption on training one data point for Le-net5 at a device is randomly drawn between 1×10^{-4} Joules and 5×10^{-4} Joules [21, 96]. The value in each figure is the mean of the results out of 50 network instances of the same size, and the running time is obtained based on a machine with a 3.79GHz AMD Ryzen 5 CPU.

To evaluate the performance of the proposed algorithms, we first compared the proposed algorithm against `FedAvg` [93] - a classic federated learning algorithm where we randomly select K devices to participate in the model training at each round. We also compare the proposed algorithm against algorithm `FedProx` in [73] and a recent algorithm `FedGrac` in [133].

We then devised a heuristic algorithm `Heur`, which chooses K devices participating in training greedily at each round. Within each iteration, algorithm `Heur` always chooses the device with the highest local loss from those yet-to-be-chosen devices as one participant at the current round, provided that the device has adequate energy to upload its local model to the server directly, or transmits its local model to its relay partner device. This procedure continues until no more devices can be added or there are K chosen devices already. Similarly, let `Heur-S` be a heuristic that adds devices with the maximum number of sampled points for each round training for the special energy-aware, D2D-assisted federated learning problem. `Heur-S` selects top-K devices with the maximum number of sampled data points to send their local models to the server. It then calculates the number of sampled data points of those not yet-to-be chosen devices if they upload their local models, using the top K chosen devices.

5.5.2 Performance of different algorithms for the special energy-aware, D2D-assisted federated learning problem

We first evaluated the performance of different algorithms for the special energy-aware D2D-assisted federated learning problem under the i.i.d data distribution. To ensure that the dataset at each device follows the i.i.d distribution, the data points with identical label are evenly distributed among devices [150]. Fig. 5.3 showed the convergence curves of the five comparison algorithms in the default setting. It can be seen that the global loss of all mentioned algorithms decreases with the growth on the number T of training rounds, and the decreasing rate on the global loss of `Algorithm` 12 is faster and more steady compared with the other four algorithms. The rationale behind is that `Algorithm` 12 makes use of more sampled data points for training at each round that leads to a lower variance on the training result, while `FedGrac` does not consider the energy capacity of devices and communication constraints on the server, it samples fewer data points for model training, which leads to a poor performance compared with `Algorithm` 12 and `Heur-S`. Consequently, `Algorithm` 12 converges to a lower global loss than those of the other four comparison algorithms, and the final global loss of `Algorithm` 12 is 15.4% lower than that of

algorithm `Heur-S`.

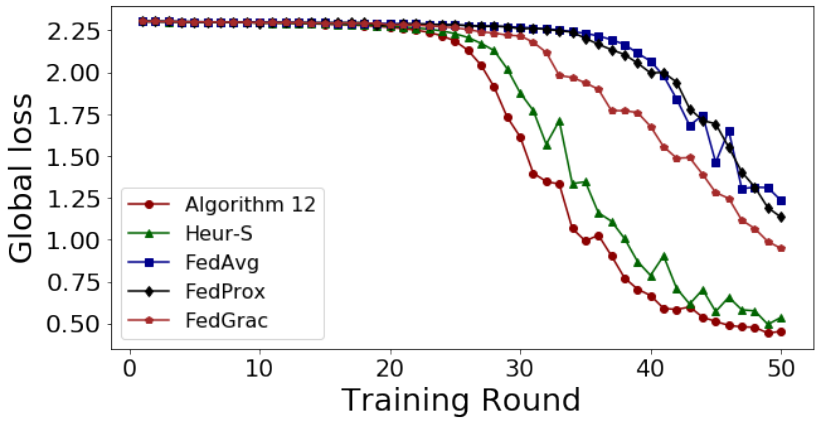


Figure 5.3: Convergence of different algorithms for the special energy-aware D2D-assisted federated learning problem with i.i.d. data and $T = 50$.

5.5.3 Performance of different algorithms for the general energy-aware D2D-assisted federated learning problem

We then studied the performance of different algorithms for the energy-aware, D2D-assisted federated learning problem under the non-i.i.d data distribution. As the training data does not follow the i.i.d distribution any more, it can be seen from Fig. 5.4 that all comparison algorithms converge slower than that of themselves for the case with the i.i.d distribution, and the final global loss is higher compared with the one for the special case. Among the five algorithms, `Algorithm` 13 has the minimum final global loss. The rationale is that `Algorithm` 13 allows more devices to participate in each round training, and always chooses devices with the highest estimated local losses. Although `FedGrac` is able to mitigate the negative effect of non-i.i.d. data distribution, it only allows at most K devices to participate in each round training, thereby resulting in a higher global loss in comparison with that of `Algorithm` 13.

5.5.4 Impact of parameters on the algorithm performance

We finally investigated the impact of parameters on the performance of different algorithms for the energy-aware D2D-assisted federated learning problem. We start by evaluating the impacts of important parameters on the special case of the problem as follows.

The impact of network size $|V|$: With the increase in network size, K ($= 20\% \cdot |V|$) increases, too. Fig. 5.5 illustrates the performance of the five mentioned algorithms with different network sizes. It can be seen from Fig. 5.5 (a) that `Algorithm` 12 is superior to the other algorithms. It is noted that all the five algorithms have lower global losses with the increase in the number K of devices that communicate with the server simultaneously. The rationale behind this is that more devices are able to participate in training at each round. Fig. 5.5 (b) shows the average running time of

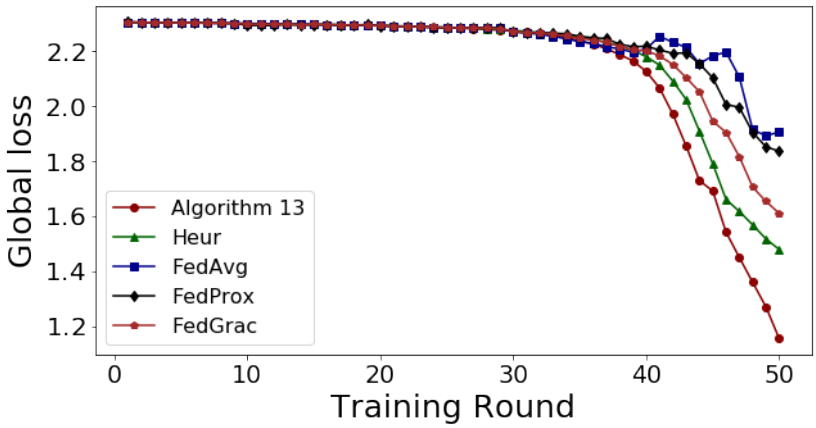
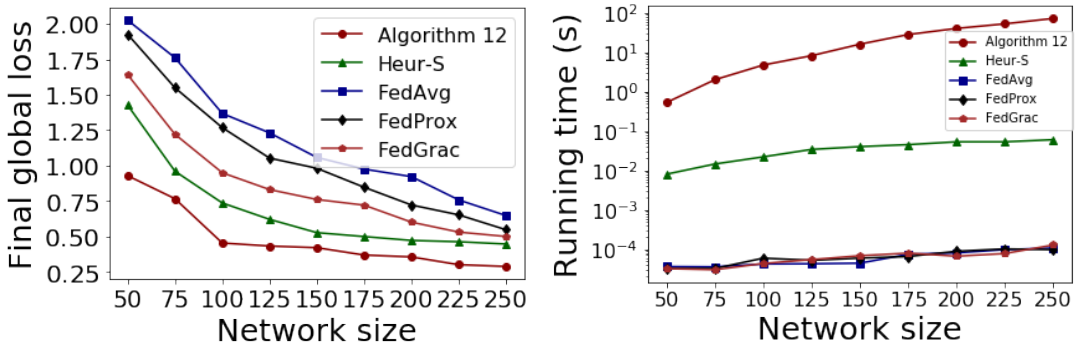


Figure 5.4: Convergence of different algorithms for the energy-aware D2D-assisted federated learning problem when $\tau = 1$ and $T = 50$.

different algorithms per training round. Although `Algorithm` 12 takes the longest running time among the five comparison algorithms, the global loss it delivers is the lowest among the five algorithms.

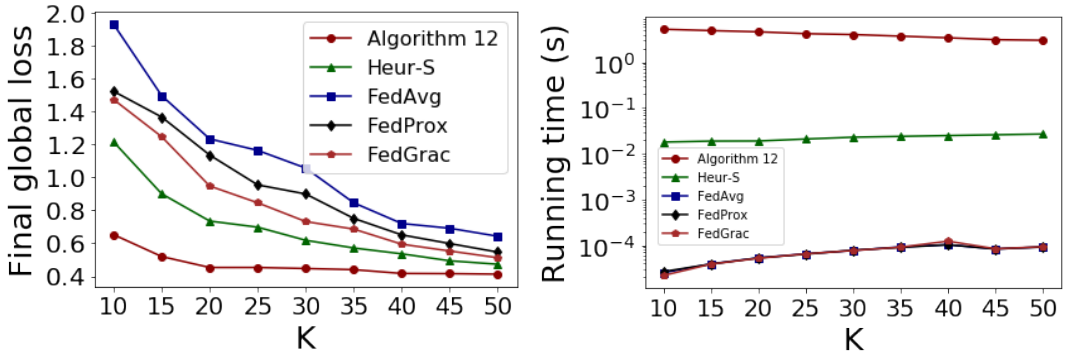


(a) The final global losses of different algorithms (b) Average running time of algorithms per training round

Figure 5.5: Performance of different algorithms for the special D2D-assisted federated learning problem by varying network size.

The impact of K: We varied the value of K from 10 to 50 while fixing the network size at 100. Fig. 5.6 (a) depicts the final global losses of different algorithms. It can be seen that `Algorithm` 12 has the lowest final global loss. Specially, when $K = 50$, the final global loss by `Algorithm` 12 is 12.78% lower than that by `Heur-S`. This can be justified by that `Algorithm` 12 achieves better decision of the destination of devices to ensure the total number of sampled data points is maximized. Fig. 5.6 (b) depicts the running times of the comparison algorithms. With the growth of K, the number of dummy nodes in the auxiliary graph decreases, leading to a decrease in the running time of `Algorithm` 12.

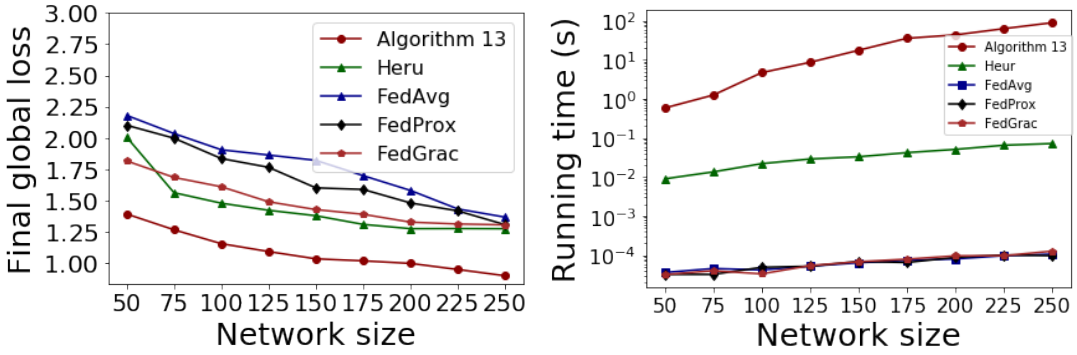
The rest is to study the impact of important parameters on the performance of the comparison algorithms for the energy-aware D2D-assisted federated learning problem as follows.



(a) The final global losses of different algorithms (b) Average running time of algorithms per training round

Figure 5.6: Performance of different algorithms for the special D2D assisted federated learning problem by varying the number K of devices communicating with the edge server simultaneously.

The impact of network size $|V|$: We analyzed the impact of network size by varying it from 50 to 250. Fig. 5.7 (a) shows that the final global losses of all algorithms decrease with the growth in network size. When $|V| = 250$, the final global loss of for `Algorithm` 13 is 35% lower than itself when $|V| = 50$. Fig. 5.7 (b) depicts the running times of the five algorithms.

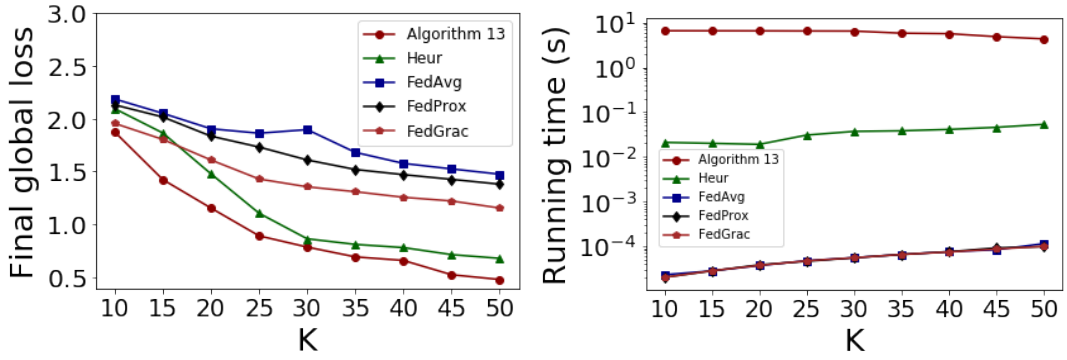


(a) The final global losses of different algorithms (b) Average running time of algorithms per training round

Figure 5.7: Performance of different algorithms for the energy-aware D2D-assisted federated learning problem by varying network size.

The impact of K: We evaluated the impact of the number K of devices that can communicate with the server simultaneously. Fig. 5.8 (a) depicts the final global losses of different algorithms by varying K from 50 to 100. When $K = 50$, the global loss by `Algorithm` 13 is 74% lower than that by itself when $K = 10$. Fig. 5.8 (b) plots the running times of the comparison algorithms. It can be seen that both the final global loss and the running time decrease with the growth of K.

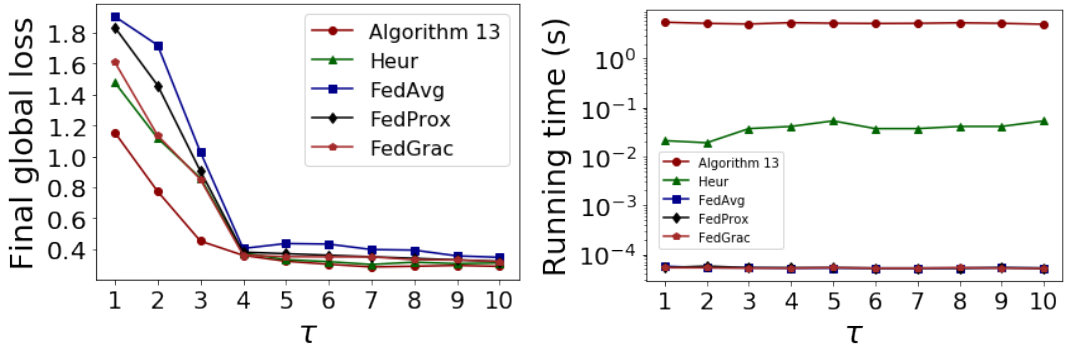
The impact of training epochs τ: We investigated the impact of the number τ of training epochs per round. To better illustrate the impact of τ, we scale up the energy budget on each device v_i to $\tau \cdot \mathcal{E}_i(t)$ at each round accordingly. Fig. 5.9 (a)



(a) The final global losses of different algorithms (b) Average running time of algorithms per training round

Figure 5.8: Performance of different algorithms for the energy-aware D2D-assisted federated learning problem by varying the number K of devices communicating with server s simultaneously.

plots the final global losses of different algorithms. It can be seen that `Algorithm` 13 has the best performance, as it has a lower final global loss when τ is no greater than 4. This means that the federated learning can train the model better with fewer training epochs when applying `Algorithm` 13. Also, when $\tau = 10$, the final global loss of algorithm `Heur` is 10.7% larger than that of `Algorithm` 13.



(a) The final global losses of different algorithms (b) Average running time of algorithms per training round

Figure 5.9: Performance of different algorithms for the D2D-assisted federated learning problem by varying τ.

5.6 Conclusion

In this chapter, we investigated the energy-aware D2D-assisted federated learning problem in an edge computing environment, by making use of the energy of neighbor devices of a certain number of devices to help their local model uploading. Under the i.i.d. training data distribution, we devised a near optimal learning algorithm for the problem. We also developed an efficient heuristic algorithm for the problem by removing the i.i.d. data distribution assumption. We finally evaluated

the performance of the proposed algorithms by experimental simulations. Simulation results demonstrated that the proposed algorithms are promising, and the performance of global loss improves 15.4% compared with that of the comparison algorithms. In our future work, we will extend the proposed algorithms by considering more than two IoT devices as a group to upload their aggregated model and examine whether they will outperform the proposed ones.

Chapter 6

Conclusions and Future Work

This chapter summarizes the contributions we made in this thesis, followed by a discussion on potential future work and research topics derived from this thesis.

6.1 Summary of Contributions

Intelligent resource allocation for IoT service provisioning in edge has been systematically studied in this thesis. We formulated novel concepts, models, frameworks and optimization algorithms to efficiently schedule the resource allocation. We devised approximation and efficient heuristic algorithms for the problem of data collection maximization in an IoT network using an energy-constrained UAV. We then solved the profit maximization problem by proposing a novel deep reinforcement learning framework for IoT service placement and an algorithm for request assignment. Thirdly, we formulated a novel federated learning framework with D2D model uploading enabled, and developed efficient algorithms to minimize losses of trained models. Moreover, we tackled the DT state staleness minimization problem by devising an approximation algorithm and a deep learning mechanism to predict the volume of generated data at IoT devices. We also considered the fidelity-aware queries on the DT by developing a cost-effective evaluation plan. In the end, we conducted comprehensive experiments to evaluate our proposed algorithms. The proposed frameworks, optimization techniques and analysis techniques may be of independent interest in many other areas, especially in the combinatorial optimization and artificial intelligence domain.

The main contributions of this thesis are summarized as follows.

- We studied the problem of maximizing data collection from IoT devices with an energy-constrained UAV. We formulated a novel data collection framework in which the UAV collects data from IoT devices within its transmission range. We then proposed two different data collection maximization problems, where the UAV fully or partially collects data from the IoT sensors. Finally, we devised efficient approximation and heuristic algorithms to solve both problems.

- We considered the DT states staleness minimization problem and the fidelity-aware query evaluation problem on DTs. We first formulated a novel staleness

minimization problem of DT states. Then, we consider a special case of the problem and devised an optimal algorithm for it. We then study the state synchronization issue for a scenario where an energy-constrained UAV is used to collect data from IoT devices (physical objects). We devised a heuristic algorithm under the assumption that the volume of generated data of IoT sensors is given. We then proposed a deep learning mechanism to predict the volume of data generated by each sensor to remove the assumption. In the end, we develop cost-efficient query evaluation plans for fidelity-aware queries built upon the DT data.

- We studied the problem of online service placement and request assignment in an MEC network with the aim to maximize the profit of service providers. To meet user service stringent delay requirements, we proposed a deep reinforcement learning-based mechanism to pre-install service instances according to the predicted most demanded services without prior knowledge of the arrival of future requests. We then devise an efficient algorithm to assign the service requests to the most appropriate cloudlets.

- We investigated the federated learning loss minimization problem in an energy-aware D2D-assisted edge environment. We first proposed a novel framework where D2D communications are enabled between devices to help their local model uploading. We then devised a near optimal learning algorithm for a special case of the problem, where the training data on devices follow i.i.d. We also studied the general case of the problem by developing an efficient heuristic algorithm.

- We conducted comprehensive simulation experiments to evaluate the performance of our proposed algorithms. Besides, we also studied on the impact of constraint parameters on the algorithms by conducting experiments under different settings. The experiments show that our algorithms are very promising, which deliver considerably better results in maximizing data collection and profit, and minimizing loss of deep neural network models and staleness of DT states, compared with existing methods.

6.2 Future Directions

In this paper, we study several problems in intelligent resource allocation for IoT services in mobile edge environments. Based on the study conducted in this thesis, there are several possible research topics worth further investigating.

First, the thesis assumes that all resources are available and functioning normally, but it is possible that edge computing systems experience failures or resource shortages. In order to achieve fault tolerance, replicas of service instances can be deployed in cloudlets. Due to the inadequate resources of cloudlets, we need to determine the number and the location of replicas wisely to meet QoS and reliability requirements. In the future, we will study on how to design intelligent resource

allocation mechanisms to provide IoT services while taking failures into consideration. For example, we can utilize deep learning methods to predict the occurrence of instance failures in advance and prepare the redundant replica and backup accordingly.

Second, the thesis only considers stationary scenarios where IoT devices are fixed. However, many IoT devices, such as vehicles, move at the most time in practice. The mobility of IoT devices physically increases the distance between the devices and the service instances, which potentially violates the latency requirement as most IoT services have stringent end-to-end delay requirements. Therefore, we will consider how to provide intelligent IoT services to IoT devices with mobility as a future topic. Initially, we will research on prediction of the future trajectory of IoT devices via artificial intelligence techniques, such as multi-armed bandit [15] or generative adversarial networks [30]. Based on the predicted trajectory, we will design the optimal trajectory of UAV to collect data to synchronize moving IoT devices with their digital twins. We will also propose algorithms to migrate IoT service instances to serve IoT service requests of moving IoT devices.

Thirdly, despite the fact that this thesis considers some techniques of artificial intelligence, it mainly focuses on the applications of general deep neural networks in mobile edge environments. However, some artificial intelligence techniques are not based on deep neural networks, for example, multi-armed bandit [15] and genetic algorithms [47], which means some optimization methods proposed in this thesis may not be applicable to these techniques. Moreover, with the advancement of artificial intelligence, there are more and more emerging deep learning techniques, such as generative adversarial networks [30], and graph neural networks [136]. These state-of-the-art techniques introduce potential performance improvement to current intelligent resource allocation frameworks and even solve more complex optimization problems. Future work will investigate how to incorporate the above artificial intelligence techniques into mobile edge environments, so that more AI applications can be better hosted by mobile edge networks.

Fourthly, although there exists long latency between devices and remote data centers, remote data centers provided sufficient resources to admit IoT service requests. However, cloud computing is not considered in this thesis. Therefore, future research could study on a three-layer hybrid framework, where IoT services are hosted at remote data centers and edge servers simultaneously. To this end, we will propose brand new algorithms to solve problems with different optimization objects when admitting IoT requests and providing AI applications in such hybrid edge networks.

Finally, most algorithms proposed in the thesis are centralized executed, which means a centralized scheduler is needed to obtain the knowledge of the whole system. One possible research topic is to develop distributed algorithms, which have several benefits: (1) The running time of the algorithms will be significantly reduced. (2) The system will be more resilient to single point of failures.

Bibliography

1. N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, Mobile edge computing: a survey. *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450 – 465, Feb. 2018. (cited on page 1)

2. A. Abdallah, M. M. Mansour, and A. Chehab. Power control and channel allocation for D2D underlaid cellular networks. *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 3217 – 3234, 2018.

3. S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani. A survey on federated learning: the journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476-5497, 2021. (cited on page 4)

4. A. Al-Hourani, S. Kandeepan, and S. Lardner. Optimal LAP altitude for maximum coverage. *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, 2014. (cited on page 23)

5. A. Ahmed and E. Ahmed. A survey on mobile edge computing *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India, 2016, pp. 1-8, doi: 10.1109/ISCO.2016.7727082. (cited on page 2)

6. M. Babaeizadeh, I. Frosio, S. Tyree,J. Clemons, and J. Kautz. Reinforcement learning through asynchronous advantage actor-critic on a GPU. [Online] Available: arXiv:1611.06256. 2016 Nov 18.

7. E. Baccour et al., Pervasive AI for IoT applications: a survey on resource-efficient distributed artificial intelligence. *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2366-2418, 2022. (cited on page 86)

8. N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. *Proc. 36th Annu. ACM Symp. Theory Comput.* (STOC), pp. 166–174, 2004. (cited on page 4)

9. M. Bastopcu and S. Ulukus. Age of information for updates with distortion: Constant and age-dependent distortion constraints. *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2425 – 2438, 2021. (cited on pages 27, 31, 32, 33, 34, and 55)

10. M.T. Beck, M. Werner, S. Feld, and S. Schimper. Mobile edge computing: a taxonomy. *Proc. of the Sixth International Conference on Advances in Future Internet*, pp. 48–55, 2014. (cited on page 10)

(cited on page 2)

11. H. Binol, E. Bulut, K. Akkaya, and I. Guvenc. Time optimal multi-UAV path planning for gathering its data from roadside units. *Proc of 88th Vehicular Technology Conference* (VTC-Fall), IEEE, pp. 1–5, 2018. (cited on page 8)

12. A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. in *Proc. 44th Annu. IEEE Symp. Found. Comput. Sci.* (FOCS), pp.46–55, 2003. (cited on pages 27 and 55)

13. B.M. Brown. Great expectations: The theory of optimal stopping. *Journal of the Royal Statistical Society: Series A (General)*, vol.135, no.4, pp.610-610, 1972. (cited on page 12)

14. G. Boss, P. Malladi, D. Quan, L. Legregni, and H. Hall. Cloud computing. *IBM white paper 321*, pp.224-231 (cited on page 1)

15. D. Bouneffouf, I. Rish and C. Aggarwal, Survey on applications of multi-armed and contextual bandits *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-8, 2020. (cited on page 123)

16. C. Chen et al. Synchronize only the immature parameters: communication-efficient federated learning by freezing parameters adaptively. *IEEE Transactions on Parallel and Distributed Systems*, doi: 10.1109/TPDS.2023.3241965. (cited on page 17)

17. M. Chen, W. Liang, and Y. Li. Data collection maximization for UAV-enabled wireless sensor networks. *Proc of 29th ICCCN'20*, IEEE, 2020. (cited on page 9)

18. M. Chen, W. Liang, and S. Das. Data collection utility maximization in wireless sensor networks via efficient determination of UAV hovering locations. *Proc of 19th International Conference on Pervasive Computing and Communications* (PerCom'21), IEEE, 2021. (cited on page 9)

19. M. Chen, W. Liang, and J. Li. Energy-efficient data collection maximization for UAV-assisted wireless sensor networks. *Proc of IEEE Wireless Communications and Networking Conference* (WCNC'21), IEEE, 2021. (cited on page 9)

20. M. Chen, H. V. Poor, W. Saad and S. Cui. Convergence time optimization for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457 – 2471, 2021. (cited on pages 15 and 16)

21. M. Chen and Z. Yang, W. Saad, C. Yin, H.V. Poor, and S. Cui. A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269 – 283, 2020. (cited on pages 94, 113, and 115)

22. S. Chen, Y. Xu, H. Xu, Z. Jiang and C. Qiao, Decentralized federated learning with intermediate results in mobile edge computing. *IEEE Transactions on Mobile Computing*, 2022, doi: 10.1109/TMC.2022.3221212. (cited on page 16)

23. X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795 – 2808, 2016.

24. X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005-4018, 2019. (cited on page 14)

25. Y. J. Cho, J. Wang, and G. Joshi. Client selection in federated learning: convergence analysis and power-of-choice selection strategies. [Online]. Available: arXiv:2010.01243, 2020. (cited on page 111)

26. N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Technical Report 388*, Graduate School of Industrial Administration, CMU, 1976. (cited on pages 35 and 39)

27. R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, vol.100, no.4, pp.162-66, 2006. (cited on pages 80 and 81)

28. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2009. (cited on page 81)

29. L. Corneo, C. Rohner and P. Gunningberg. Age of information-aware scheduling for timely and scalable internet of things applications. *Proc. of INFOCOM'19*, pp. 2476 – 2484, IEEE, 2019. (cited on page 10)

30. A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath. Generative adversarial networks: an overview. *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53-65, 2018, doi: 10.1109/MSP.2017.2765202. (cited on page 123)

31. R. Cziva, C. Anagnostopoulos, and D. P. Pezaros. Dynamic, latency-optimal VNF placement at the network edge. *Proc. of INFOCOM'18*, pp. 693-701, IEEE, 2018. (cited on page 12)

32. S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya. Edge intelligence: the confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457– 7469, Aug. 2020. (cited on page 4)

33. Digital twin market worth 15.66 billion USD by 2023, (n.d.). *https://www.marketsandmarkets.com/PressReleases/digital-twin.asp*, accessed, Feb., 2022. (cited on page 5)

34. Digital twin | Siemens. *https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465*, accessed, Feb., 2022. (cited on page 5)

35. C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Zomaya, and D. Gramoli. Federated learning over wirelss networks: convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, vol. 29, no, 1, pp. 398 – 409, 2021. (cited on pages 15 and 16)

36. H. Djigal, J. Xu, L. Liu and Y. Zhang. Machine and deep learning for resource allocation in multi-access edge computing: a survey. *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2449– 2494, 2022. (cited on page 4)

37. R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic. Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin. *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692 – 4707, 2019. (cited on pages 5 and 10)

38. J. Du, L. Zhao, J. Feng, X. Chu, and F. R. Yu. Economical revenue maximization in cache enhanced mobile edge computing. *Proc. of ICC'18*, pp. 1-6, IEEE, 2018. (cited on page 13)

39. M. B. Ghorbel, D. Rodriguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar. Energy efficient data collection for wireless sensors using drones. *Proc of 87th VTC Spring*, pp. 1–5, IEEE, 2018. (cited on page 8)

40. I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. *MIT press*, 2016. (cited on page 4)

41. Q. Guo, J. Peng, W. Xu, W. Liang, X. Jia, Z. Xu, Y. Yang and M. Wang. Minimizing the longest tour time among a fleet of UAVs for disaster area surveillance. *IEEE Transactions on Mobile Computing*, vol.24, no.7, pp. 2451 - 2465, 2022. (cited on page 9)

42. H. O. Hassan, S. Azizi, and M. Shojafar. Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments. *IET communications*, vol. 14, no. 13, pp. 2117-2129, 2020. (cited on page 13)

43. S. Hayat, E. Yanmaz, and R. Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2624-–2661, 2016. (cited on page 8)

44. T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein. It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. *Proc. of ICDCS'18*, pp. 365-375, IEEE, 2018. (cited on page 13)

45. Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin. Software-defined networks with mobile edge computing and caching for smart cities: a big data deep reinforcement learning approach. *IEEE Communications Magazine* vol. 55, no. 12, pp. 31-37, 2017. (cited on page 14)

46. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, vol.9, no.8, pp.1735-80, 1997. (cited on pages 57 and 58)

47. J.H. Holland. Genetic algorithms. *Scientific american*, vol.267, no.1, pp.66-73. 1992. (cited on page 123)

48. C. Hu, W. Bao, D. Wang, and F. Liu. Dynamic adaptive DNN surgery for inference acceleration on the edge. *Proc. of INFOCOM'19*, pp. 1423 – 1431, IEEE, 2019.

49. L. Huang, S. Bi, and Y. -J. A. Zhang. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581-2593, 2020. (cited on page 14)

50. S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto. Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training with Non-IID Private Data. *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2021.3070013, 2021. (cited on page 15)

51. E. Jack. Maximum matching and a polyhedron With O,1-vertices. *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55 – 56, 1965. (cited on pages 102, 103, 110, and 114)

52. M. Jia, W. Liang, and Z. Xu. QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc. of MSWiM'17*, pp.109–116, ACM, 2017. (cited on pages 88 and 89)

53. P. Kayal and J. Liebeherr. Distributed service placement in fog computing: An iterative combinatorial auction approach. *Proc. of ICDCS'20*, pp.2145-2156, IEEE, 2020. (cited on page 12)

54. L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong. Digital-twin-enabled 6G: vision, architectural trends, and future directions. *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74 – 80, 2022. (cited on pages 5 and 10)

55. J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, vol.23, no.3, pp.462–466, 1952. (cited on page 84)

56. G. Lai, W.C. Chang, Y. Yang, H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 95 – 104, 2018. (cited on page 66)

57. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278 – 2324, 1998. (cited on page 115)

58. Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou. Profit-driven service request scheduling in clouds. *Proc. of CCGrid'10*, pp. 15-24, ACM, 2010. (cited on page 11)

59. J. Li, W. Liang, M. Huang, and X. Jia. Reliability-aware network service provisioning in mobile edge-cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, vol.31, no.7, pp.1545-58, 2020. (cited on page 13)

60. J. Li, W. Liang, Y. Li, Z. Xu, and X. Jia. Delay-aware DNN inference throughput maximization in edge computing via jointly exploring partitioning and parallelism. *Proc. of LCN'21*, pp. 193-200, IEEE, 2021. (cited on page 73)

61. J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo. Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism. *IEEE Transactions on Mobile Computing*, vol.22, no.5, pp.3017-3030, 2023. (cited on pages 73 and 113)

62. J. Li, W. Liang, and Y. Ma. Robust service provisioning with service function chain requirements in mobile edge computing. in *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2138-2153, 2021. (cited on pages 13 and 88)

63. J. Li, W. Liang, Z. Xu, X. Jia, and W. Zhou. Service provisioning for multi-source IoT applications in mobile edge computing. *ACM Transactions on Sensor Networks*, vol. 18, no. 2, Article 17, pp. 17:1 – 17:25, 2022, (cited on pages 65 and 66)

64. J. Li, W. Liang, W. Xu, Z. Xu, Y. Li, and X. Jia. Service home identification of multiple-source IoT applications in edge computing. *IEEE Transactions on Services Computing*, vol.16, no.2, pp. 1417 – 1430, 2023. (cited on page 65)

65. J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, and J. Zhao. Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no.5, pp. 1199-1212, 2022. (cited on page 11)

66. J. Li, W. Liang, W. Xu, Z. Xu, and J. Zhao. Maximizing the quality of user experience of using services in edge computing for delay-sensitive IoT applications. *Proc. of MSWiM'20*, ACM, 2020. (cited on page 21)

67. J. Li, W. Liang, Z. Xu, and W. Zhou. Provisioning virtual services in mobile edge computing for IoT applications with multiple sources. *Proc. of LCN'20*, pp. 42 – 53, IEEE, 2020. (cited on page 21)

68. Y. Li, W. Liang, and J. Li. Profit maximization for service placement and request assignment in edge computing via deep reinforcement learning. *Proc of The 24th*

International Conference on Modeling, Analysis and Simulation of Wireless and Mobile System (MSWiM'21), ACM, 2021.

69. Y. Li, W. Liang, J. Li, X. Cheng, D. Yu, A. Zomaya, and S. Guo. Energy-constrained D2D assisted federated learning in edge computing. *Proc of 25th MSWiM'22*, Oct., ACM, 2022.

70. Y. Li, W. Liang, W. Xu, and X. Jia. Data collection of IoT devices using an energy-constrained UAV. *Proc of the 34th IEEE International Parallel and Distributed Processing Symposium* (IPDPS), IEEE, pp.644–653, 2020.

71. Y. Li, W. Liang, W. Xu, Z. Xu, X. Jia, Y. Xu and H. Kan. Data collection maximization in IoT sensor networks via an energy-constrained UAV. *IEEE Transactions on Mobile Computing*, Vol.22. No.1 pp. 159 – 174, 2023. (cited on page 66)

72. Q. Li et al. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347-3366, 2023. (cited on page 4)

73. T. Li, A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith. Federated optimization in heterogeneous networks *Proceedings of Machine learning and systems*, vol. 2, pp. 429 – 450, 2020. (cited on page 115)

74. W. Liang, W. Xu, X. Ren, X. Jia, and X. Lin. Maintaining large-scale rechargeable sensor networks perpetually via multiple mobile charging vehicles. *ACM Trans. Sensor Netw.*, vol.12, no.2, Article 14, 2016. (cited on page 7)

75. W. Liang, Z. Xu, W. Xu, J. Shi, G. Mao, and S. Das. Approximation algorithms for charging reward maximization in rechargeable sensor networks via a mobile charger. *IEEE/ACM Transactions on Networking*, vol.25, no.5, pp.3161 – 3174, 2017. (cited on pages 7, 56, and 57)

76. Y. Liang, W. Xu, W. Liang, J. Peng, X. Jia, Y. Zhou, and L. Duan. No-redundant information collection in rescue applications via an energy-constrained UAV. *IEEE Internet of Things Journal*, vol.6, no.2, pp.2945–2958, 2019. (cited on page 8)

77. X. Lin, J. Wu, J. Li, W. Yang and M. Guizani. Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach. *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2402-2416, 2023. (cited on page 10)

78. J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu. On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543-553, 2017. (cited on page 13)

79. N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. *Proc. of ICDCS'17*, pp. 372-382, IEEE, 2017. (cited on page 14)

80. Y. Liu, H. Yu, S. Xie, and Y. Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168, 2019. (cited on page 14)

81. Q. Liu, H. Zeng, and M. Chen. Minimizing AoI with throughput requirements in multi-path network communication. *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1203-1216, 2022. (cited on page 10)

82. R. Lu, et al., Auction-based cluster federated learning in mobile edge computing systems. it IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 4, pp. 1145 – 1158, 2023. (cited on page 16)

83. Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2276 – 2288, 2021. (cited on page 10)

84. Y. Lu, S. Maharjan and Y. Zhang. Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16219 – 16230, 2021. (cited on page 10)

85. Z. Lv, J. Hao, and Y. Guo. Energy minimization for MEC-enabled cellular-connected UAV: trajectory optimization and resource scheduling. *Proc. of INFOCOM'20*, IEEE, 2020. (cited on pages 98 and 113)

86. Y. Ma, W. Liang, and W. Xu. Charging utility maximization in wireless rechargeable sensor networks by charging multiple sensors simultaneously. *IEEE/ACM Transactions on Networking*, vol.26, no.4, pp.1591–1604, 2018. (cited on page 7)

87. Y. Ma, W. Liang, Z. Xu, and S. Guo. Profit maximization for admitting requests with network function services in distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, vol.30, no.5, pp.1143-57, 2018. (cited on pages 13 and 88)

88. A. Macariousm. Assessment of distilBERT performance on named entity recognition task for the detection of protected health information and medical concepts. *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pp. 158 – 167, 2020.

89. P. Mach and Z. Becvar. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628 – 1656, 2017. (cited on page 1)

90. S. Madakam, R. Ramaswamy, and S. Tripathi. Internet of Things (IoT): a literature review. *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164, 2015. (cited on page 3)

91. Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief. A survey on mobile edge computing: the communication perspective. *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322 – 2358, 2017. (cited on page 1)

92. J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. *Proc. of NSDI'14* (USENIX), 2014. (cited on page 88)

93. B. McMahan, M. Eider, D. Ramage, S. Hampson, and A. Blaise. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*, pp. 1273 – 1282. PMLR, 2017. (cited on page 115)

94. M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang. Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930-1941, 2019. (cited on page 14)

95. R. Minerva, G. M. Lee and N. Crespi. Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785 – 1824, 2020. (cited on pages 5 and 10)

96. T. Mohammed, C. Joe-Wong, R. Babbar, and M. D. Francesco. Distributed inference acceleration with adaptive DNN partitioning and offloading. *Proc. of INFOCOM'20*, IEEE, pp. 854 – 863, 2020. (cited on page 115)

97. M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Mobile Internet of Things: can UAVs provide an energy-efficient mobile architecture? *Proc of IEEE Globecom'16*, 2016. (cited on pages 8 and 23)

98. M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications. *IEEE Transactions on Wireless Communications*, vol.16, no.11, pp. 7574–7589, 2017. (cited on page 8)

99. Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, T. Qiu, B. Hu, R. Y. K. Kwok. Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1277-1292, 2021. (cited on page 13)

100. T. Nishio, and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *Proc. of ICC'19*, IEEE, 2019. (cited on pages 15 and 16)

101. A. Paul, D. Freund, A. Ferber, D. Shmoys, and D. Williamson. Prize-collecting TSP with a budget constraint. *Proc. 25th Annu. Eur. Symp. Algorithms (ESA)*, Wadern, Germany: Schloss Dagstuhl-Leibniz- Zentrum fuer Informatik, 2017, pp. 62:1–62:14.

102. Phantom 4 pro V2 Specification. [Online]. Available: https://www.dji.com/au/phantom-4-pro-v2/info#specs, 2018. (cited on page 41)

103. L. L. Pilla. Scheduling algorithms for federated learning with minimal energy consumption. *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 4, pp. 1215–1226, 2023. (cited on pages 15 and 17)

104. K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas Joint service placement and request routing in multi-cell mobile edge computing networks. *Proc. of INFOCOM'19*, pp. 10-18, IEEE, 2019. (cited on page 73)

105. Predix platform. *https://www.ge.com/digital/iiot-platform*, accessed Feb, 2022. (cited on page 5)

106. H. Qi and A. Gani, Research on mobile cloud computing: review, trend and perspectives, *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, Bangkok, Thailand, 2012, pp. 195-202, doi: 10.1109/DICTAP.2012.6215350. (cited on page 1)

107. D. Reinsel, J. Gantz, and J. Rydning. The digitization of the world for edge to core. IDC, Framingham, MA, USA, white paper #: US44413318, pp. 28, 2018. (cited on page 3)

108. A. Samanta and Z. Chang. Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint. *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3864-3872, 2019. (cited on page 13)

109. M. Samir, S. Sharafeddine, C. Assi, T. Nguyen, and A. Ghrayeb. UAV trajectory planning for data collection from time-constrained IoT devices. *IEEE Transactions on Wireless Communications*, vol.19, no.1, pp. 34–46, 2020. (cited on page 9)

110. O. Said, and M. Masud. Towards internet of things: Survey and future vision. *International Journal of Computer Networks*, vol.5, no.1, pp. 1–17, 2017. (cited on page 3)

111. S. N. Shirazi, A. Gouglidis, A. Farshad and D. Hutchison. "The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog From a Security and Resilience Perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017. (cited on page 2)

112. D. Sikeridis, E.E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou. Wireless powered Public Safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency. *Journal of Network and Computer Applications*, vol.123, no.1, pp. 69–79, 2018. (cited on page 8)

113. Q. Sun, P. Lu, W. Lu, and Z. Zhu. Forecast-assisted NFV service chain deployment based on affiliation-aware VNF placement. *Proc. of GLOBECOM'16*, pp. 1-6, IEEE, 2016. (cited on page 12)

114. W. Sun, H. Zhang, R. Wang, and Y. Zhang. Reducing offloading latency for digital twin edge networks in 6G. *IEEEE Trans. Veh. Technol.*, vol 69, no.10, pp 12240 – 12251, 2020. (cited on pages 5 and 10)

115. X. Sun and N. Ansari. EdgeIoT: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22 – 29, 2016. (cited on page 3)

116. Y. Sun, S. Zhou, and D. Gundz. Energy-aware analog aggregation for federated learning with redundant data. *Proc. of ICC'20*, IEEE, 2020. (cited on pages 15 and 16)

117. M. F. Sohail, C.Y. Leow, and S. Won. Energy-efficient non-orthogonal multiple access for UAV communication system. *IEEE Transactions on Vehicular Technology*, vol. 68, no.11, pp.10834–10845, 2019. (cited on page 23)

118. R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, pp. 1057-1063, 2020. (cited on page 86)

119. M. Tang and V. W. S. Wong. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985-1997, 2022. (cited on page 14)

120. Z. Tang, S. Shi, B. Li and X. Chu, GossipFL: a decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2023, doi: 10.1109/TPDS.2022.3230938. (cited on page 17)

121. Y. Tao, S. Cui, W. Xu, H. Yin, D. Yu, W. Liang, and X. Cheng. Byzantine-resilient federated learning at edge. *IEEE Transactions on Computers*, 2023, doi: 10.1109/TC.2023.3257510. (cited on page 17)

122. J. Theunissen, H. Xu, R. Y. Zhong, and X. Xu. Smart AGV system for manufacturing shopfloor in the context of industry 4.0. *Proc of 25th International Conference on Mechatronics and Machine Vision in Practice* (M2VIP), pp.1 – 6, 2018. (cited on page 40)

123. P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: a survey. *European Journal of Operational Research*, vol. 209, pp.1 — 10, 2011. (cited on pages 27 and 29)

124. S. Voghoei, NH. Tonekaboni , JG. Wallace, and Arabnia HR. Deep learning at the edge. *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, pp. 895-901, 2018. (cited on page 4)

125. C. Wang, Z. Cai, and Y. Li. Sustainable blockchain-based digital twin management architecture for IoT devices. *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6535-6548, 2023. (cited on page 10)

126. X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen. Convergence of Edge Computing and Deep Learning: A Comprehensive Survey.

IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 869-904, 2020, doi: 10.1109/COMST.2020.2970550. (cited on page 4)

127. X. Wang, Z. Ning, S. Guo, M. Wen and V. Poor. Minimizing the age-of-critical-information: An imitation learning-based scheduling approach under partial observations. *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3225-3238, 2022. (cited on page 10)

128. S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung. Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002-1016, 2017. (cited on page 73)

129. Z. Wang, L. Duan, and R. Zhang. Adaptive deployment for UAV-aided communication networks. *IEEE Transactions on Wireless Communications* vol. 18, no. 9, pp. 4531 – 4543, 2019. (cited on page 113)

130. S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong. Network-aware optimization of distributed learning for fog computing. *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2019 – 2032, 2021. (cited on pages 16 and 107)

131. S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205 – 1221, 2019. (cited on pages 15, 16, 96, 102, 104, and 115)

132. F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun. Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach. *Proc. of INFOCOM'20*, pp.2499-2508, IEEE, 2020. (cited on page 84)

133. F. Wu, et al. From deterioration to acceleration: a calibration approach to rehabilitating step asynchronism in federated optimization. *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 5, pp. 1548 – 1559, May 2023. (cited on page 115)

134. Q. Wu, et al. HiFlash: communication-efficient hierarchical federated learning With adaptive staleness control and heterogeneity-aware client-edge association. *IEEE Transactions on Parallel and Distributed Systems*, 2023, doi: 10.1109/TPDS.2023.3238049. (cited on pages 15 and 16)

135. Y. Wu, K. Zhang, and Y. Zhang. Digital twin networks: a survey. *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13789 – 13804, 2021. (cited on page 5)

136. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, 2021, doi: 10.1109/TNNLS.2020.2978386. (cited on page 123)

137. J. Xu, L. Chen, and P. Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. *Proc. of INFOCOM'18*, IEEE, 2018. (cited on page 11)

138. W. Xu, W. Liang, X. Jia, H. Kan, Y. Xu, and X. Zhang. Minimizing the maximum charging delay of multiple mobile chargers under the multi-node energy charging scheme. *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1846-1861, 2021. (cited on page 7)

139. W. Xu, W. Liang, H. Kan, Y. Xu, and X. Zhang. Minimizing the longest charge delay of multiple mobile chargers for wireless rechargeable sensor networks by charging multiple sensors simultaneously. *Proc of 39th IEEE Intl Conf. of Distributed Computing Systems*, pp.881–890, 2019. (cited on page 7)

140. W. Xu, W. Liang, X. Lin, and G. Mao. Efficient scheduling of multiple mobile chargers for wireless sensor networks. *IEEE Trans. Veh. Technol.*, vol.65, no.9, pp.7670–7683, 2016. (cited on page 7)

141. W. Xu, W. Liang, Z. Xu, J. Peng, D. Peng, T. Liu, X. Jia, and S. Das. Approximation algorithms for the generalized team orienteering problem and its applications. *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 176-189, Feb. 2021. (cited on page 7)

142. Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, vol.18, no.11, pp.2672-85, 2018. (cited on pages 13, 88, and 89)

143. Z. Xu, D. Li, W. Liang, W. Xu, Q. Xia, P. Zhou, O. Rana, and H. Li. Energy or accuracy? near-optimal user selection and aggregator placement for federated learning in MEC. *IEEE Transactions on Mobile Computing*, 2023, doi: 10.1109/TMC.2023.3262829. (cited on page 17)

144. Z. Xu, D. Zhao, W. Liang, O. Rana, P. Zhou, M. Li, W. Xu, H. Li, and Q. Xia. HierFedML: aggregator placement and UE assignment for hierarchical federated learning in mobile edge computing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 34. no. 1, pp. 328 – 345, 2023. (cited on pages 15 and 17)

145. Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu. QoS-aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks. *ACM Transactions on Sensor Networks*, vol.16, no.3, pp.1-27, 2020. (cited on page 12)

146. Z. Xu, W. Ren, W. Liang, W. Xu, Q. Xia, P. Zhou, and M. Li. Schedule or wait: age-minimization for IoT big data processing in MEC via online learning. *Proc of INFOCOM'22*, pp. 1809 – 1818, IEEE, 2022. (cited on pages 10 and 11)

147. W. Yang, X. Xiang, Y. Yang, and P. Cheng. Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT. *IEEE*

Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1884 – 1893, 2023. (cited on page 15)

148. Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935 – 1949, 2021. (cited on page 15)

149. Z. Yang, Y. Cui, X. Wang, Y. Liu, M. Li, and Z. Zhang. Towards maximal service profit in geo-distributed clouds. *Proc. of ICDCS'19*, pp.442-452, IEEE, 2019. (cited on page 13)

150. N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani. Hybrid-FL for wireless networks: cooperative learning mechanism using non-IID data. *Proc. of ICC'20*, pp. 1 – 7,2020. (cited on page 115)

151. A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue. FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework. *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080-5096, 2019, doi: 10.1109/JIOT.2019.2896311. (cited on page 13)

152. C. Zhan and Y. Zeng. Completion time minimization for multi- UAV-enabled data collection. *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4859 – 4872, 2019. (cited on page 8)

153. C. Zhan, Y. Zeng, and R. Zhang. Energy-efficient data collection in UAV enabled wireless sensor network. *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, 2017. (cited on page 8)

154. J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. A Akerkar. Adaptive federated learning on Non-IID data with resource constraint. *IEEE Transactions on Computers*, doi: 10.1109/TC.2021.3099723, 2021. (cited on pages 15, 16, and 95)

155. J. Zhang, Z. Li, W. Xu, J. Peng, W. Liang, Z. Xu, X. Ren, and X. Jia. Minimizing the number of deployed UAVs for delay-bounded data collection of IoT devices. *Proc of INFOCOM'21*, pp. 1-10, 2021. (cited on page 9)

156. S. Zhang, L. Wang, H. Luo, X. Ma, and S. Zhou. AoI-delay tradeoff in mobile edge caching with freshness-aware content refreshing. *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5329 – 5342, 2021. (cited on pages 10 and 11)

157. J. Zhang, Z. Li, W. Xu, J. Peng, W. Liang, Z. Xu, X. Ren, and X. Jia. Minimizing the number of deployed UAVs for delay-bounded data collection of IoT devices. *Proc. of INFOCOM'21*, IEEE, 2021. (cited on page 66)