Flow-Time Minimization for Timely Data Stream Processing in UAV-Aided Mobile Edge Computing

ZICHUAN XU and HAIYANG QIAO, School of Software, Dalian University of Technology, Dalian, P. R. China
WEIFA LIANG, Department of Computer Science, City University of Hong Kong, Hong Kong, P. R. China
ZHOU XU, School of Software, Dalian University of Technology, Dalian, P. R. China
QIUFEN XIA, International School of Information Science and Engineering, Dalian University of Technology, Dalian, P. R. China
PAN ZHOU, The Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, P. R. China
OMER F. RANA, Physical Sciences and Engineering College, Cardiff University, Cardiff, United Kingdom
WENZHENG XU, College of Computer Science, Sichuan University, Chengdu, P. R. China

Unmanned Aerial Vehicles (UAVs) have gained increasing attention by both academic and industrial communities, due to their flexible deployment and efficient line-of-sight communication. Recently, UAVs equipped with base stations have been envisioned as a key technology to provide 5G network services for mobile users. In this article, we provide timely services on the data streams of mobile users in a UAV-aided Mobile Edge Computing (MEC) network, in which each UAV is equipped with a 5G small-cell base station for communication and data processing. Specifically, we first formulate a flow-time minimization problem by jointly caching services and offloading tasks of mobile users to the UAV-aided MEC with the aim to minimize the flow time, where the flow time of a user request is referred to the time duration from the request issuing time point to its completion point, subject to resource and energy capacity on each UAV. We then propose a spatial-temporal learning optimization framework. We also devise an online algorithm with a competitive ratio for the problem based upon the framework, by leveraging the round-robin scheduling and dual fitting techniques. Finally, we evaluate the performance of the proposed algorithms through experimental simulation. The simulation results demonstrate that the proposed algorithms outperform their comparison counterparts, by reducing the flow time no less than 19% on average.

The work by Z. Xu and Q. Xia was supported by the National Natural Science Foundation of China (NSFC) (grants 62172068 and 62172071) and Shandong Provincial Natural Science Foundation (grants ZR2023LZH013 and ZR2023LZH008). The work by W. Liang was supported by a grant from City University of Hong Kong (project 9380137/CS). The work by W. Xu was supported by the NSFC (grant 62272328), and the work by P. Zhou was supported in part by the NSFC (grant 61972448).

Authors' addresses: Z. Xu, H. Qiao, Z. Xu, and Q. Xia, School of Software Technology, Dalian University of Technology No. 321 Tuqiang Street, Economy & Technology Development Zone, Dalian, Liaoning, People's Republic of China, 116620; e-mails: z.xu@dlut.edu.cn, qiaohy@mail.dlut.edu.cn, 21917040@mail.dlut.edu.cn, qiufenxia@dlut.edu.cn; W. liang, Department of Computer Science, City University of Hong Kong 83 Tat Chee Avenue Kowloon Tong, Kowloon KLN HK 999077; e-mail: weifa.liang@cityu.edu.hk; P. Zhou, The Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology Luoyu Road 1037 Wuhan, People's Republic of China 430074; e-mail: panzhou@hust.edu.cn; O. F. Rana, Physical Sciences and Engineering College, Cardiff University, Cardiff Wales UK CF10 3AT; e-mail: RanaOF@cardiff.ac.uk; W. Xu (Corresponding author), College of Computer Science, Sichuan University, Jiang'an Campus, Sichuan University Chuanda Road, Shuangliu County, Chengdu, People's Republic of China, 610207; e-mail: wenzheng.xu@scu.edu.cn.

ACM 1550-4859/2024/03-ART58
https://doi.org/10.1145/3643813

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing design and evaluation methods**; • **Computing methodologies** → *Machine learning;*

Additional Key Words and Phrases: Mobile edge computing, unmanned aerial vehicles, service caching and task offloading, online algorithm, machine learning

ACM Reference Format:
Zichuan Xu, Haiyang Qiao, Weifa Liang, Zhou Xu, Qiufen Xia, Pan Zhou, Omer F. Rana, and Wenzheng Xu. 2024. Flow-Time Minimization for Timely Data Stream Processing in UAV-Aided Mobile Edge Computing. *ACM Trans. Sensor Netw.* 20, 3, Article 58 (March 2024), 28 pages. https://doi.org/10.1145/3643813

1 INTRODUCTION

Recently, **Unmanned Aerial Vehicles (UAVs)** carrying small-cell base stations with computing units (e.g., neural network accelerators and FPGAs) are considered a novel **Mobile Edge Computing (MEC)** technique in 5G and beyond 5G networks [13, 31, 35]. UAVs with small-cell base stations can serve mobile users in various emergent events [32], such as earthquakes and storm floods, by dynamically dispatching UAVs to extend the coverage areas of terrestrial macro base stations. For example, the Wing Loong from China was used to provide emergency communications in the 2021 Henan flood [38]. Besides communication services, in such critical incidents, UAVs can also implement emergency AI services, such as victim identification via on-time video stream processing, by carrying AI accelerators and FPGAs, when terrestrial base stations are out of the range [11, 13, 35].

Data stream processing of mobile users in emergent events is one of fundamental network services in UAV-aided MEC, such as continuous object recognition and video streaming processing. Such data streams usually require timely processing. Otherwise, the emergent events may not be resolved. For example, in rescue services of critical incidents, if the captured videos by UAVs are not identified in a timely manner, the victims may not be rescued or the critical incident may worsen. In other words, the responsiveness to critical incidents is crucial in rescue services. The *flow time*, defined as the time from data generation to its completion of data processing, is emerging as a novel and effective metric to capture the timely responsiveness of UAV-based rescue services. In this article, we study the flow-time minimization problem for processing data streams in UAV-aided MEC, by jointly caching services from ground base stations and offloading user requests to the cached services in UAVs equipped with small-cell base stations.

Minimizing the flow time for user requests with continuous data streams in UAV-aided MEC is challenging:

— First, the arrival times and the data volumes of user requests are usually uncertain, which however determines the network performance. In particular, the request data volumes have spatial and temporal correlations and can be impacted by various side information, such as festivals and weather. Thus, precise spatial-temporal predictions that embed such side information play a vital role in minimizing the flow time of user requests. Otherwise, requests in some locations may experience prohibitively long processing delays. How to jointly consider dispatching UAVs to strategic locations, cache services, and offload tasks of mobile users is challenging.
— Second, the performance of service caching and task offloading for data stream processing depends on intertwining factors, such as energy statuses of UAVs, distances between UAVs, and resource capacities of UAVs. How to jointly consider dispatching UAVs to strategic locations, cache services, and offload tasks of mobile users is challenging.

— Third, given data stream processing requests issued by users in different locations, minimizing the flow time of one request may lead to the starvation of another request with long processing times or latency. How can we find a fine-grained tradeoff between the fairness and the flow time among different user requests?

To the best of our knowledge, we are the first to investigate the flow-time minimization problem in UAV-aided MEC under the assumption of uncertain arrival times and data volumes of user requests, by joint service caching and task offloading for timely data stream processing. Although there are studies on leveraging UAVs to collect data for **Internet of Things (IoT)** networks [3, 12, 19, 30, 51, 54, 55], continuous data stream processing is largely ignored. Further, none of these existing studies aimed to minimize the flow time of requests with uncertain arrival times and data volumes. In addition, studies on service caching and task offloading have been extensively explored in conventional MEC [18, 24, 42, 43, 45, 58] instead of UAV-aided MEC. The existing methods and algorithms of the aforementioned studies thus cannot be directly applied to the problem of this study.

The main contributions of this work are as follows:

— We formulate a novel optimization problem of the flow-time minimization problem in UAV-aided MEC, by formulating an **Integer Linear Programming (ILP)** solution, assuming that the request arrivals and data volumes of requests are given.
— We propose a spatial-temporal learning optimization framework with uncertain request arrivals and data volumes of user requests, and an online algorithm with a provable competitive ratio for the problem by leveraging the **Round-Robin (RR)** scheduling and dual fitting techniques.
— We evaluate the performance of the proposed algorithms by extensive simulations, and results show that the performance of our algorithms outperform an existing study by reducing the flow time at least 19% on average.

The rest of the article is organized as follows. Section 2 summarizes related studies. Section 3 introduces the system model and defines the problem. Section 4 presents an exact solution to the offline version of the problem. Section 5 details a learning-based optimization framework. Section 6 evaluates the performance of the proposed algorithms, and Section 7 concludes the article.

2 RELATED WORK

Although there are extensive studies on task offloading and service caching for mobile users in an MEC network, most of them focused on conventional MEC environments on the ground [6, 18, 24, 37, 41, 43, 45, 58]. Aerial service caching and task offloading for data stream processing have been ignored. For example, Jeong et al. [18] studied a problem of incremental offloading for AI applications, by designing an algorithm that determines the optimal deep neural network partition and offloading sequences. Tout et al. [37] developed a two-level cost-effective and intelligent resource-aware optimization model to determine what computations to offload for serving multiple mobile devices and reduce the remote execution costs.

UAV-aided MEC is gaining much attention due to the high flexibilities provided by UAVs. However, most existing studies have focused on communications relay, content caching, data collection, or sensory coverage [3, 12, 19, 21, 22, 30, 51, 54, 55, 59]. Joint service caching and task offloading in UAV-aided MEC with flying base stations have not been explored in the afore-mentioned studies. For example, Xu et al. [44] investigated the problem of minimizing the deployment cost of UAVs to collect data from IoT networks. Chen et al. [7, 8] studied the problem of using an energy-constrained UAV to assist sensor data collection in IoT networks to maximize the volume of data collected by the UAV. Du et al. [12] explored the problem of minimizing the energy consumption of a UAV by jointly optimizing the scheduling, resource allocation, and hovering duration of the UAV, and designed an efficient iterative algorithm for the problem. Mozaffari et al. [30] focused on the coverage and deployment of UAV small cells in a low altitude platform, by considering the influence of interference between multiple UAV small cells on coverage performance. Azizi et al. [3] raised an issue of the revenue maximization in 5G networks with the ground and aerial base stations by exploring heterogeneity of edge servers and base stations. Kong et al. [19] studied the problem of scheduling a number of UAVs to serve mobile users in an area to optimize system load, latency, and throughput. Zhong et al. [54] aimed to maximize the throughput of a UAV-aided and self-organized device-to-device (D2D) network by considering uncertainties of user locations and channel models. However, it seems that the preceding studies are unlikely to be applicable directly or indirectly to data stream processing in this study, as the problems dealt are fundamentally different. For example, if UAVs serve as a communication relay, the computing resource allocation usually does not need to be considered. In addition, the data collection in UAV-enabled IoT networks assumes that UAVs only need to collect data without processing the collected data. In contrast, in UAV-aided MECs, the data streams continuously arrive into the system and need to be processed by network services implemented in the dispatched UAVs.

Meanwhile, there are several studies on provisioning data processing services in UAV-aided MECs [9, 10, 12, 23, 34, 39, 40, 46, 49, 50, 52, 57]. However, they do not consider the timely processing of data streams from mobile users. In addition, they do not find a non-trivial tradeoff between the fairness on different user request processing and the responsiveness of network services. For example, Chen et al. [9] investigated the problem of optimizing the quality of experience of wireless devices in UAV networks. Yu et al. [49] studied the problem of minimizing the weighted sum of service delays of all devices and the energy consumption of a single UAV. In contrast, Lin et al. [23] considered the coordination of individual rationality and social benefits in a UAV-assisted MEC system. Du et al. [12] designed a novel UAV-enabled wirelessly powered mobile edge system to minimize the energy consumption of a UAV. Zhang et al. [57] proposed a data offloading protocol for an MEC with UAVs by considering both computing and communication resources. Xu et al. [47] aimed to minimize the completion time of tasks in a UAV-aided MEC network via allowing both partial and binary offloading schemes. However, they did not consider the processing of data streams, and the placement of services and the fairness of task offloading were not considered. Although Zhang et al. [53] aimed to optimize the delay of processing data streams, they did not consider the service caching and task offloading in MEC.

In summary, the aforementioned studies on service caching and task offloading in conventional MECs did not consider the timely processing of data streams of users [6, 18, 24, 43, 45, 50], whereas the studies [3, 12, 19, 30, 51, 54, 55] on UAV-aided networks neither considered service caching and task offloading. In addition, existing works of service provisioning in UAV networks ignored the responsiveness of applications [9, 12, 49, 52, 57]. In contrast, in this article, we explore the flow-time minimization problem for processing data streams of users in UAV-aided MEC via service caching and task offloading, by striving for the finest balancing between the fairness and the flow time among different requests of mobile users.

3 PRELIMINARIES

In this section, we first describe the system model, notations, and notions, then define the problems precisely.

3.1 System Model

We consider a UAV-aided MEC $G = (V, E)$, where macro base stations on the ground and UAVs in the air jointly provide services for mobile users in a given area. Note that macro base stations

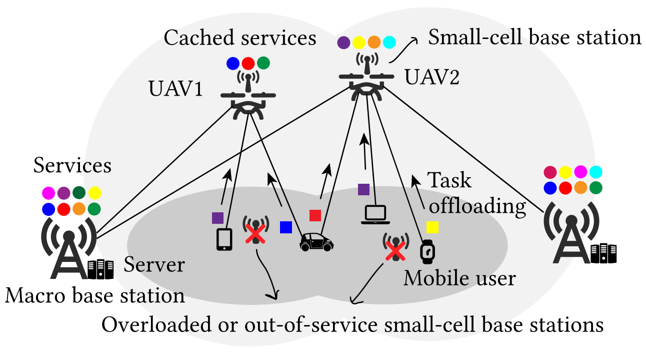


Fig. 1. An example of the UAV-aided MEC $G = (V, E)$.

can be congested when the network traffic is surging or out of reach of users in emergent events. We thus consider UAVs attached with small-cell base stations as an extension of macro base stations. Specifically, UAVs attached with small-cell base stations can fly to a location to alleviate the pressure of macro base stations in emergent events, such as network failures or hotspot traffic [11, 13, 35], as shown in Figure 1. Let $\mathcal{BS}$ be a set of macro base stations, and $bs_o \in \mathcal{BS}$ is a macro base station with $1 \leq o \leq |\mathcal{BS}|$. Denote by $\mathcal{U}$ a set of UAVs in the area, and let $u \in \mathcal{U}$ be a UAV. Then, $V = \mathcal{BS} \cup \mathcal{U}$. The small-cell base station attached to UAV u has a computing capability that can implement services required by mobile users. Mobile users can access the services provided by the MEC network via connecting to small-cell base stations carried by UAVs in $\mathcal{U}$. E is the set of communication links between base stations, UAVs, and mobile users.

3.2 Data Stream Processing, Aerial Service Caching, and Task Offloading

In UAV-based rescue services, each UAV usually senses the critical incident by continuously sending data streams for identification. For instance, a video stream may need to be processed in a timely manner to monitor the current status of the incident or identify potential victims of the incident. As such, to timely process the video streams in such incidents, AI services are cached services from remote data centers into macro base stations at network edge [1, 5, 14]. Then, users can detect critical incidents (mudslides or fire hazards) with ultra-low latency, which enables users to respond more promptly to these emergencies.

In this work, we consider that AI services have been deployed into macro base stations of UAV-aided MEC. Such services require processing continuous data streams. Note that the macro base stations may be out of reach of such requests, due to the destruction by unexpected incidents, such as mudslides or earthquakes. When the macro base stations are out of reach due to the incidents, the timely processing of data streams may not be possible. Instead, we can cache such services from macro base stations to UAVs that carry small-cell base stations, which is referred to as *aerial service caching* [33, 52]. We thus assume that macro base stations do not receive data streams from mobile users directly, and the cached services in UAVs can process the data streams directly. Thus, the UAVs with cached services then can fly to a given area to provide timely data stream processing for mobile users in the area. Let $\mathcal{S}$ be the set of services and S_i a service in $\mathcal{S}$. Assuming that the storage of each UAV is adequate, each service has its container image pre-stored in the small-cell base station of each UAV. If S_i is cached into a UAV u, a *service instance* will be created by running the container with the pre-stored image of S_i.

Let r_j be a request of a mobile user that would like to monitor the status of a critical incident. Each r_j streams an amount of data to a UAV for processing. Denote by D_j the total volume of

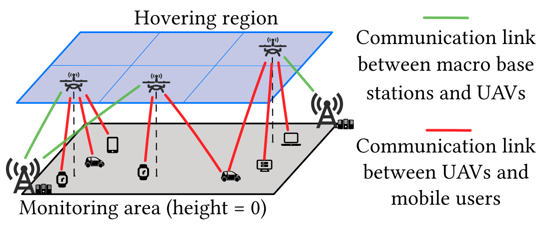

Hovering region
Communication link between macro base stations and UAVs
Communication link between UAVs and mobile users
Monitoring area (height = 0)

Fig. 2. The UAV hovering model, where the solid lines represent communication links while the dotted lines mean projections of the locations of UAVs in the air to the location on the ground.

data of r_j. Assume that time is divided into equal time slots, and each time slot t lasts τ time units. Each r_j arrives into the system dynamically, and its arrival time is denoted by a_j. Once r_j is scheduled, it can offload its data stream to the MEC continuously until all of its data D_j are processed. Meantime, the system starts processing the offloaded data. Similar to Qu et al. [33], we assume that the processing result of each request is small in size, and the time of transmitting such results can be ignored. For example, some face recognition applications only need to return a name. Let C_j be the time slot when the system completes the processing of request r_j. The *flow time* of request r_j is defined by

$$C_j - a_j. \tag{1}$$

Offloading r_j requires its service S_i to be cached in a UAV. However, caching the service of r_j permanently in a UAV may not be possible due to the resource capacities on a UAV. We thus need to update the state data generated while processing data by the cached instance of S_i with its original instance in a macro base station. We assume that the volume of state data of S_i is $\phi_j D_j$, where ϕ_j is a given constant with $0 < \phi_j \leq 1$.

The arrival time and data volumes of each request r_j are uncertain in UAV-aided MEC, considering that events related to the data streams are bursty. For instance, it is impossible to know when pedestrians will appear in front of a vehicle in autonomous driving scenarios. However, these parameters do determine the performance of UAV-aided MEC. In particular, if a request with high resource demand is offloaded to a UAV with an inadequate amount of available resources, the request may not be implemented successfully. We thus need to accurately predict the arrival time and data volumes of each request.

3.3 UAV Dispatching, Hovering, and Data Processing

In critical incidents, data stream processing usually lasts for long periods. For example, the object recognition in a flooded area usually needs to be performed continuously based on the live feed of videos, to identify trapped people. We thus assume that continuous processing of data streams lasts for tens of minutes or even hours. We further assume that each UAV is dispatched to a hovering location for processing data streams one time, and it is called back from its hovering location until it finishes the data stream processing or its energy level is low.

There are infinite potential hovering locations for UAVs in the sky. Following standard practices [19, 21, 22], to make the problem tractable, we partition the hovering region of UAVs into a finite number of identical squares, according to both the transmission range of UAVs and the number of user requests in the area. We make sure all requests can be responded to if each square has a single UAV, as shown in Figure 2. Let Q be the number of squares of the area. The center of the qth square is denoted by coordinates (x_q, y_q, H_q), where q is in the range of $[1, Q]$ and H_q is the hovering height of the UAV in square q. Note that each UAV can receive requests within its communication range (not just within its hovering square) and process its data.

We consider dispatching UAVs in $\mathcal{U}$ to Q squares of a hovering region to process the data streams of mobile users. Each dispatched UAV u can process the data as long as it receives the first unit of data of request r_j. It must be mentioned that the processing of such requests is fundamentally different with conventional data processing. Specifically, conventional data processing usually begins after receiving all data of a request. In contrast, data stream processing timely processes data while receiving it. For example, augmented reality applications typically require frame by frame rendering of video streams. Therefore, if r_j is scheduled to be processed in a number of time slots after its arrival time slot a_j, its data stream has to start being forwarded to the selected UAV in its scheduled time slots. To ensure the fairness of resource usages by different user requests and to avoid starvation of requests, we assume that each UAV adopts a preemption-enabled operating system, such as real-time Linux. Each request r_j can be preempted by other requests and resumed for later execution [27], and its data can be processed at different time slots since its arrival. Let z_{jtq} be the amount of request r_j's data that is scheduled for processing in the UAV of square q at time slot t, where z_{jtq} is an integer with $1 \le z_{jtq} \le D_j$. In addition, considering that requests of mobile users are highly dynamic, UAVs will be dynamically redispatched based on predicted data volumes of user requests.

3.4 Energy Models of UAVs

UAVs have limited battery power. Each dispatched UAV is recalled back for charge until it has done its service or its energy. As such, most of the energy consumed by each UAV is due to its activities when hovering in its dispatched location. We thus assume that each UAV preserves enough energy capacity for dispatching, redispatching, and returning for recharge, which is denoted by EN^{other} and given as *a priori*. Note that unlike data collection in IoT, UAVs can finish the data collection of an IoT device quickly, where dispatching and redispatching energy is the major energy consumption. Since the UAVs spend most of their time in data stream processing while hovering, the energy consumption due to dispatching and redispatching thus is not the major consumption. Let EN^{total} be the total energy capacity of a UAV. The energy consumption due to activities when hovering in its dispatched location cannot be greater than $EN^{total} - EN^{other}$, which include communicating with mobile users and macro base stations, processing offloaded requests, and hovering.

Each UAV $u \in \mathcal{U}$ receives a data stream from mobile users and updates the states of its cached services to their original instances of the services in macro base stations. Let $e^{rcv}_{j,q}(z_{jtq})$ be the energy consumption of the UAV in square q due to receiving the amount z_{jtq} of data from request r_j, then

$$e^{rcv}_{j,q}(z_{jtq}) = (z_{jtq}/R_{j,q}) \cdot P^r_u, \tag{2}$$

where P^r_u is the data reception power of a UAV, and $R_{j,q}$ is the achieved data rate of the wireless channel between the mobile user of r_j and the UAV in square q [57]. Recall that UAV u hovers at the center (x_q, y_q, H_q) of square q, and the distance $d_{j,q}$ can be calculated by

$$d_{j,q} = \sqrt{(x_j - x_q)^2 + (y_j - y_q)^2 + {H_q}^2},$$

where $(x_j, y_j, 0)$ is the location of the mobile user of r_j. Let $e^{upd}_{q,o}(z_{jtq})$ be the energy consumption of the UAV in square q due to transmitting the state data of the cached service instance of service S_i from the UAV in square q to its nearest macro base station bs_o at time slot t, then

$$e^{upd}_{q,o}(z_{jtq}) = (\phi_j \cdot z_{jtq}/R_{q,o}) \cdot P^t_u, \tag{3}$$

where P^t_u is the transmitting power of UAV u, and $R_{q,o}$ is the achieved data rate via the wireless channel between the UAV in square q and base station bs_o that can be obtained similar to the calculation of $R_{j,q}$.

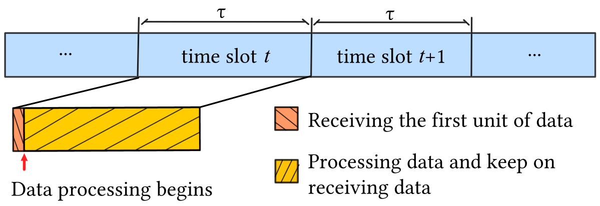

τ
τ
···
time slot t
time slot t+1
···
Receiving the first unit of data
Processing data and keep on receiving data
Data processing begins

Fig. 3. Data transmission and processing at each time slot t.

Following existing studies [15, 26], the energy consumption of a UAV per computing unit is proportional to its workload and the maximum power per computing unit. The workload of processing data is proportional to the amount of data [46, 57], and the workload of r_j thus is $b_j \cdot z_{jtq}$, where b_j is a given constant. Let $e^{cmp}(z_{jtq})$ be the amount of energy consumed due to processing an amount z_{jtq} of request r_j's data in the UAV of square q, then

$$e^{cmp}(z_{jtq}) = \tau\big((\xi \cdot b_j \cdot z_{jtq}/\tau)P^{max} + P'\big),$$

where ξ is a given parameter that is used to calculate the access rate of computing units of a UAV as shown in the work of Hong and Kim [15], P^{max} is the maximum power of all computing units of a UAV, and P' is the sum of the idle power and the leakage power of a UAV.

Denote by ζ the energy consumption rate per unit time on hovering. The energy consumption of each UAV on hovering thus is

$$e^{hov}(t) = \tau \cdot \zeta. \tag{4}$$

3.5 Transmission and Resource Consumption Models

Once the UAV in square q receives the first unit D_{unit} of data of request r_j in each time slot t, the data processing procedure will start immediately, as shown in Figure 3. For example, an application of object detection can start processing a stream of images once it receives the first few images. The time that can be used for data processing is $\tau - \frac{D_{unit}}{R_{j,q}}$, if r_j is scheduled in time slot t, where $R_{j,q}$ is the data transmission rate between the mobile user of r_j and the UAV in square q. Following many existing studies [18, 48], the data of most requests can be processed within a time of δ by assigning amount η of a computing resource to process a unit amount of its data. To speed up the processing, the UAV can assign more computing resources to process the data. Specifically, if UAV u wants to process z_{jtq} amount of data of r_j within $\tau - (D_{unit}/R_{j,q})$ time in time slot t with length τ, the amount of computing resource needed is

$$C(z_{jtq}) = \eta \cdot z_{jtq}\big(\delta/(\tau - (D_{unit}/R_{j,q}))\big). \tag{5}$$

3.6 Problem Definition

Given a UAV-aided MEC $G = (V, E)$ providing AI services for a set R of user requests, we aim to enhance the responsiveness of AI services by minimizing the total flow time of all requests. However, this may not be fair for some requests that are forced to wait for prohibitively long times before being scheduled. Thus, to strive for a fine balance between the fairness and the flow time among requests, we adopt the l_k-norm of flow time of requests, defined by

$$l_k = \left(\sum_{r_j \in R} (C_j - a_j)^k\right)^{1/k}, \tag{6}$$

where k is a fixed integer in the range of $[1, 3]$.

Given a set R of user requests with a given time horizon T and the data volumes of user requests, *the offline flow-time minimization problem in UAV-aided MEC* is to dispatch UAVs in $\mathcal{U}$ to the Q squares of a monitoring area to serve requests in a given time horizon T, by caching the services in $\mathcal{S}$ to the dispatched UAVs and offloading the tasks of requests to the UAVs, such that the l_k-norm flow time of all admitted requests is minimized, subject to the computing capacity CR and energy capacity EN^{total} on each UAV.

Assuming that requests arrive into the system dynamically without the knowledge of future arrivals, and the data volumes of user requests are uncertain, *the online flow-time minimization problem in UAV-aided MEC* is to jointly predict the data volumes of user requests and dispatch UAVs in $\mathcal{U}$ to the Q squares of a monitoring area to serve requests in a given time horizon T, by caching the services in $\mathcal{S}$ to the dispatched UAVs and offloading the tasks of requests to the UAVs, such that the l_k-norm flow time of all admitted requests is minimized, subject to the computing capacity CR and energy capacity EN^{total} on each UAV.

The symbols used in this article are summarized in Table 1.

4 ILP FORMULATION FOR THE OFFLINE FLOW-TIME MINIMIZATION PROBLEM

We now consider the offline flow-time minimization problem given request arrival time and data volume. The offline flow-time minimization problem is to minimize the l_k-norm of the total flow time of all requests, subject to computing and energy capacities on each UAV. Note that each request is completed only when all of its data is processed. In each time slot t, a portion data of each request r_j may be offloaded to a UAV for processing. We thus use z_{jtq} to denote the amount of data of request r_j offloaded to the UAV in square q for processing at time slot t, and $\mathbf{z} = \{z_{jtq} \mid \forall t, r_j \in R, q \in [1,Q]^+\}$. Let $x_{t,q}$ be a binary decision variable that shows whether there is a UAV hovering in square q at time slot t. The objective of the problem then is to

$$\textbf{ILP:} \min_{\mathbf{z}} \left(\sum_{r_j \in R} (C_j - a_j)^k \right)^{1/k}, \tag{7}$$

subject to

$$\sum_{t \geq a_j} \sum_{q} z_{jtq} = D_j, \forall r_j \in R \tag{8}$$

$$\sum_{q=1}^{Q} x_{t,q} = |\mathcal{U}|, \forall t \tag{9}$$

$$\sum_{j:t \geq a_j} \eta \cdot z_{jtq} \cdot \delta / \left(\tau - \frac{D_{unit}}{R_{j,q}} \right) \leq x_{t,q} \cdot CR, \forall t, q \in [1,Q]^+ \tag{10}$$

$$\sum_{t, r_j \in R} e^{rcv}_{j,q}(z_{jtq}) + e^{upd}_{q,o}(z_{jtq}) + e^{cmp}(z_{jtq}) \leq x_{t,q} EN, \tag{11}$$

$$z_{jtq} \in [1, D_j]^+, \tag{12}$$

where $EN = EN^{total} - EN^{other}$.

Constraint (8) says that request r_j can only be implemented after its arrival and all of its data has to be processed. Constraint (9) guarantees that there are $|\mathcal{U}|$ UAVs in all squares in any time slot. Constraint (10) indicates that the computing resource capacity of UAV u cannot be violated, where the left-hand side of Equation (10) includes the time used for transmitting an amount z_{jtq} of r_j to the UAV in square q. Intuitively, if the transmission of a unit amount of data D_{unit} takes

Table 1. Symbols

Notation	Definition
$G = (V, E)$	A UAV-aided MEC with a set $\mathcal{BS}$ of macro base stations with each base station denoted by bs_o
$\mathcal{U}$ and u	A set of UAVs, a UAV in $\mathcal{U}$
$\mathcal{S}$ and S_i	A set of services and a service in $\mathcal{S}$
r_j and D_j	A user request and the data volume of r_j
D_{max}, D_{unit}	The maximum data volume of requests and a unit data of a request
t, τ	The tth time slot and the number of time units per time slot
a_j, C_j	The time slot when r_j arrives into the system and the time slot when the system completes implementing r_j
ϕ_j, Q	The ratio of the volume of state data to the volume of r_j's data, and the number of squares
(x_q, y_q, H_q), $(x_j, y_j, 0)$	The location of the center of qth square and the location of the mobile user of r_j
H_q	The hovering height of the UAV in square q
z_{jtq}	The amount of r_j's data that is scheduled for processing in the UAV of square q at time slot t
$e^{rcv}_{j,q}(z_{jtq})$	The energy consumption of the UAV in square q due to receiving the amount z_{jtq} of data from r_j
$e^{upd}_{q,o}(z_{jtq})$	The energy consumption of the UAV in square q due to transmitting the state data from the UAV in square q to bs_o at time slot t
$e^{cmp}(z_{jtq})$	The energy consumption of the UAV in square q due to processing an amount z_{jtq} of data
$e^{hov}(t)$	The energy consumption of each UAV on hovering at a time slot
P^r_u, P^t_u	The data reception and transmitting powers of UAV u
$R_{j,q}$, $R_{q,o}$	The achieved data rate between the mobile user of r_j and the UAV in square q and that between the UAV in square q and macro base station bs_o
R_{min} and $R'_{\min}$	The minimum $R_{j,q}$ for mobile users in the transmission range of UAV u, and the minimum $R_{q,o}$ for all macro base stations in the transmission range of UAV u
P^t_j, B	The transmitting power of the mobile user of r_j, and the wireless bandwidth assigned to a UAV
β	The channel gain at the referencedistance 1 m
$d_{j,q}$	The distance between the UAV in square q and the mobile user of r_j
b_j	A given constant that is used to calculate the workload of processing data
ξ	A given parameter that is used to calculate the access rate of computing units of a UAV
P^{max} and P'	The maximum power and the sum of the idle power and the leakage power of all computing units of a UAV
ζ, δ and η	The energy consumption rate on hovering, the data of most requests can be processed within a time of δ by assigning amount η of computing resource to process a unit amount
$C(z_{jtq})$	The amounts of computing resource needed if UAV u wants to process z_{jtq} amount of data within $\tau - (D_{unit}/R_{j,q})$
l_k	The l_k norm of flow time, where k is a fixed integer in the range of [1, 3]
CR, EN^{total}	The computing and energy capacities on each UAV
$x_{t,q}$	A binary decision variable, whether there is a UAV hovering in square q at time slot t
p	A time period with multiple time slots
M, N	The number of rows and columns of cells of an area, respectively
$cell_{m,n}$, $\mathcal{D}^p_{m,n}$	The cell located at the row m and column n, the total data volume of requests from $cell_{m,n}$ in period p
w_o, $\hat{\mathcal{D}}'_q(p)$	The number of periods looking ahead, and the predicted data volume of square q in the beginning of period p
$\mathcal{U}_d$ and u'_l	A set of dispatched UAVs and the lth virtual UAV of UAV u
$C'(D_{max})$ and L	The computing capacity of each virtual UAV and the number of virtual UAVs of UAV u
z'_{jt}	An amount of r_j's data that is scheduled for processing by a virtual UAV at time slot t
e'_{unit}	The maximum energy consumption of transmitting, updating, and processing a unit amount of data by each virtual UAV
$\omega_{j,t}$	The impact on the flow time if the arrival request r_j is assigned at time slot t
$\rho_{j,t}$, ρ	A discount factor on the impact of scheduling r_j on the currently alive requests, and $\rho = \max\{\rho_{j',t'}\}$
n_t	The number of alive requests at time slot t
α_j, β_t, γ_j	The dual variables
T_o, T_u	The set of overloaded and underloaded time slots
$R_{alv}(t)$	The set of alive requests at time slot t
$R_{alv}(t, \leq a_j)$ and $R_{alv}(t, \geq a_j)$	The set of alive requests arrived prior to and after r_j at time slot t
λ	The parameter that captures the length of time of r_j's impact after its competition

longer time in time slot t, less time can be used for processing z_{jtq} amount of data. However, we can assign more computing resource to speed up the process such that the amount z_{jtq} of data is processed within time slot t [18, 48]. Constraint (11) ensures that the energy consumption of each UAV does not exceed its energy capacity. Constraint (12) guarantees that z_{jtq} is an integer in the range of $[1, D_j]$.

5 A LEARNING-BASED OPTIMIZATION FRAMEWORK FOR THE ONLINE FLOW-TIME MINIMIZATION PROBLEM

In reality, the request arrivals are not known and the data volumes of requests are uncertain. Although the ILP formulation delivers an optimal solution to the offline flow-time minimization problem, it cannot handle the dynamic request arrivals and uncertain data volumes effectively. To resolve this issue, here we propose a learning-based optimization framework for the online flow-time minimization problem, following a divide-and-conquer strategy. Specifically, we first assume that the data volumes are given and a number of UAVs have been dispatched. We propose an online algorithm to schedule requests to the UAVs to harness the dynamic request arrivals. We then devise a learning-based algorithm to dispatch UAVs with uncertain data volumes.

5.1 Online Algorithm with a Number of Dispatched UAVs and Given Data Volumes

Given a number of dispatched UAVs and the data volumes, we need to assign the dynamically arrived requests to the dispatched UAVs. Note that even given a number of dispatched UAVs and given data volumes of requests, the request arrival time may not be known in advance. The immediate scheduling of an arrived request may lead to the starvation of a small request that may arrive in a very short time. To avoid the starvation of some requests, we use the RR policy to fairly schedule requests, which gives equal sharing among UAVs to all requests. In addition, to make sure that the computing capacity of each UAV is met, we divide the resources of each UAV and ensure that each piece of computing capacity divided by the UAV can handle one request.

Specifically, we create a number of *virtual UAV copies* of each UAV u with each having the same location as u. Denote by u'_l the lth virtual UAV of UAV u. We ensure that each virtual UAV should have the sufficient resource to implement a single request. In other words, any virtual UAV has enough resource to implement the request with the maximum data volume that is transmitted via the wireless link with the minimum rate. Then, each u'_l has an amount

$$C'(D_{max}) = \eta \cdot D_{max}\big(\delta/(\tau - D_{unit}/R_{min})\big) \tag{13}$$

of computing resource, where $D_{max} = \max\{D_j \mid \forall r_j \in R\}$, and $R_{min} = \min\{R_{j,q}\}$ for mobile users in the transmission range of UAV u. Similarly, to avoid quick energy depletion, we assume that the amount of communication energy consumed by each virtual UAV u'_l is due to the communication with the farthest mobile user. In other words, for a unit amount of data, virtual UAV u'_l consumes the amount $\frac{P^r_u}{R_{min}}$ and $\frac{P^t_u}{R'_{min}}$ of energy for receiving and sending a unit amount of data, respectively, where $R_{min} = \min\{R_{j,q}\}$ for mobile users in the transmission range of UAV u, and $R'_{\min} = \min\{R_{q,o}\}$ for all macro base stations in the transmission range of UAV u. The number of virtual UAVs of UAV u is

$$L = \lfloor CR/C'(D_{max}) \rfloor. \tag{14}$$

Let z'_{jt} be the amount of data of r_j scheduled for processing by one virtual UAV at time slot t, and $\mathbf{z}' = \{z'_{jt} \mid \forall t, r_j \in R\}$. Following other studies [4, 17], we approximate the objective in ILP by

$$\mathbf{LP}: \min_{\mathbf{z}'} \ \big((t - a_j)^k/D_j + D_j^k/D_j\big) \cdot z'_{jt}, \tag{15}$$

subject to

$$\sum_{t \geq a_j} z'_{jt} \geq D_j, \ \forall r_j \in R \tag{16}$$

$$\sum_{j:t \geq a_j} z'_{jt} \leq |\mathcal{U}| \cdot L \cdot D_{max}, \ \forall t \tag{17}$$

$$\sum_{t} z'_{jt} \cdot e'_{unit} \cdot L \leq EN, \ \forall r_j \in R \tag{18}$$

$$0 \leq z'_{jt} \leq D_j, \tag{19}$$

where e'_{unit} is the maximum energy consumption of transmitting, updating, and processing a unit amount of data by each virtual UAV.

Constraint (16) ensures that all data of r_j is processed. Constraint (17) guarantees that the total volume of data dispatched to UAVs at each time slot does not exceed the total amount of data that can be processed by all UAVs. Constraint (18) says that the energy consumption of each request does not exceed the capacity of each virtual UAV.

The flow-time minimization problem then is reduced to the problem of allocating data portions of each request r_j to different time slots. The allocated data portions in each time slot are then scheduled by the RR scheduling policy with $|\mathcal{U}| \cdot L$ virtual UAVs, and the requests that are scheduled for execution and have not finished processing of all data are called *alive requests*. To enhance the service responsiveness, we adopt an immediate-dispatch rule. In other words, each request is assigned to a virtual UAV without waiting on its arrival. A negative impact of this immediate-dispatch rule is that the alive requests may have less time using the computing resource, because the RR scheduling policy is adopted. Besides, the assignment of a request has an impact on the admissions of other requests in time slot t. To minimize the flow time, requests with less impact should be preferred. Specifically, on the arrival of request r_j, it is assigned to a virtual UAV that has a minimum impact on the flow time of existing requests in the system, where the impact of r_j on the total flow time at time slot t is defined as

$$\omega_{j,t} = (\rho_{j,t} \cdot k \cdot (t - a_j)^{k-1})/n_t + k \cdot (t - a_j)^{k-1} - \epsilon, \tag{20}$$

where $\rho_{j,t}$ is a discount factor on the impact of scheduling r_j on the currently alive requests, n_t is the number of alive requests at time slot t, and ϵ is a constant.

The detailed steps of the algorithm are shown in Algorithm 1, referred to as OL. As shown in Figure 4, in the beginning of each time slot t, newly arrived and unfinished user requests are considered as to-be-scheduled requests in current time slot t, by adding them into set R_t. Then, all requests in R_t are sorted in increasing order of their impact defined in Equation (20); let the sorted set be recorded as R_t^s. Next, each request r_j in set R_t^s is assigned to a virtual UAV of UAV u according to the RR scheduling policy, where the mobile user generating request r_j is within the transmission range of UAV u. It must be mentioned that a request may be assigned to different UAVs for processing at different time slots. In particular, when a UAV is running out of energy and returning for recharge, its unfinished requests will be assigned to other UAVs in the next time slot.

5.2 A Learning-Based Online Optimization Framework

We now propose an online optimization framework based on predict-and-dispatch strategy for the problem by removing the assumption that the data volumes are given and the UAVs are already dispatched. We first predict the data volumes of user requests, then dispatch or redispatch (when UAVs are already dispatched) UAVs to squares according to the prediction results. However, if the data volumes of requests are predicted at each time slot, it may lead to frequent UAV dispatchments, thereby wasting energy on flying around. We thus let p be a period with multiple time slots and

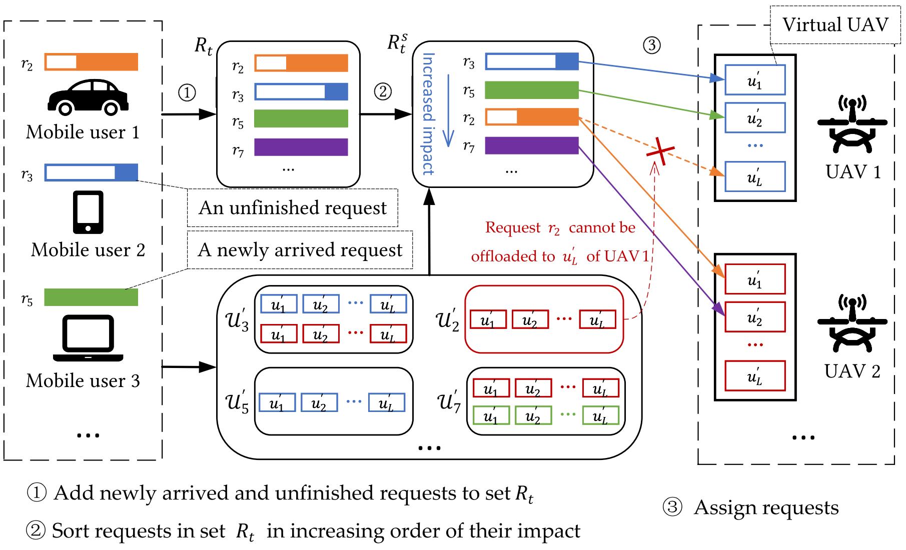


Fig. 4. An illustration of Algorithm 1.

ALGORITHM 1: OL

```
Input: $G = (V, E)$, a set $R$ of user requests that arrive into the system dynamically, a set $\mathcal{U}$ of UAVs that are already
    dispatched to $Q$ squares.
Output: Schedules for all requests in $R$.
 1: Create $L$ virtual copies of each UAV $u$ by Equation (14);
 2: $R_t \leftarrow \emptyset$;
 3: for each time slot $t$ do
 4:     Add newly arrived and unfinished requests to set $R_t$;
 5:     Sort all requests in set $R_t$ in increasing order of their impact defined in Equation (20), let $R_t^s$ be the sorted requests;
 6:     for each request $r_j$ in $R_t^s$ do
 7:         $\mathcal{U}_j' \leftarrow \emptyset$ ;
 8:         for each UAV $u$ in $\mathcal{U}$ do
 9:             if the mobile user of $r_j$ is in the transmission range of UAV $u$ then
10:                 Add all virtual UAV $u_l'$ of UAV $u$ to set $\mathcal{U}_j'$;
11:     Assign each request $r_j$ in $R_t^s$ to a virtual UAV in $\mathcal{U}_j'$, by the RR scheduling policy;
12:     All assigned requests to each virtual UAV share the same amount of resource of the UAV;
13: return A scheduling for all requests.
```

predict the data volumes of requests every period. This means that when there are fewer time slots in a period, the UAVs will be dispatched more frequently.

The data volumes of requests usually have spatial and temporal patterns due to the mobility of users. To learn such spatial and temporal patterns, our idea is to use a grayscale image to represent the accumulative data volumes of requests in a hovering region for a given time period. It is known that the convolutional neural network architecture is a powerful tool to capture spatial information, whereas the **Long Short-Term Memory (LSTM)** method [16] is capable of performing accurate time series predictions. We thus make a full customization of convLSTM [36] to predict data volumes of requests for the next few periods, by taking a series of grayscale images as inputs. The prediction results are finally used to guide the dispatchment of UAVs.

The proposed learning optimization framework is detailed as follows.

Spatial-Temporal Prediction. Mobile users in each square may form into different groups, and their data volumes are correlated in each group. We thus divide each square of the area into a

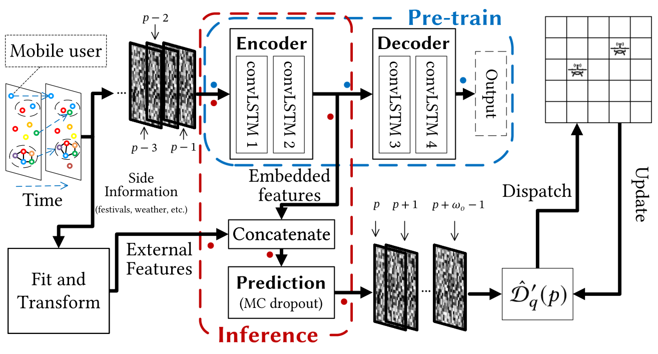


Fig. 5. The learning-based online optimization framework.

number of cells with smaller sizes. We predict the data volumes in a fine granularity of cells. Let M and N be the number of rows and columns of all cells in the area. Denote by $cell_{m,n}$ the cell located at row m and column n. In the grayscale image of a period, its pixel in row m and column n denotes the data volume in $cell_{m,n}$. Then, the gray value of the pixel in $cell_{m,n}$ is $\mathcal{D}^p_{m,n}/\max \mathcal{D}^p_{m,n}$, where $\mathcal{D}^p_{m,n}$ is the total data volume of all requests from $cell_{m,n}$ in period p.

The prediction algorithm based on a series of given grayscale images is described as follows.

Considering that the data volumes of requests are affected by many types of side information, such as festivals and weather, we need to embed these *external features* into the neural network such that a higher accuracy can be obtained. To this end, we refine the neural network in the work of Zhu and Laptev [56] by replacing the traditional LSTM with the convLSTM, as shown in Figure 5. Specifically, the training phase of the learning-based online optimization framework is divided into the following two steps. The first step is pre-training. As shown in the blue dashed box in Figure 5, we first use a series of grayscale images to train the encoder and decoder. The second step is training. To effectively utilize side information, we first encode the side information through a three-dimensional vector, and then perform fitting and transforming to obtain external features, which contains information such as weather, festivals, and assemblies that affect data volumes of requests. However, the pre-trained decoder cannot efficiently use these external features. Therefore, we concatenate the external features and embedded features obtained by the pre-trained encoder to train the neural network used for prediction, as shown in the red dashed box in Figure 5. In the inference stage, we can use a series of grayscale images and side information as inputs to predict the data volumes of requests for future periods.

To reduce the dispatch distance of UAVs, we predict the accumulative data volume of all requests in each square by looking ahead a number of periods. The rationale behind this is that prediction each period may lead to frequent redispatchments of UAVs. Let w_o be the number of time periods that we look ahead. As shown in Figure 5, the spatial-temporal prediction network will output grayscale images of the next few periods, which represent data volumes of requests. However, considering that the longer we look ahead, the lower the prediction accuracy will be, we put an exponentially decreasing weight on the predicted values of the w_o periods. We then dispatch UAVs according to the weighted sum of data volumes of w_o periods. Let $\hat{\mathcal{D}}'_q(p)$ be the predicted data volume of square q in the beginning of period p. Then,

$$\hat{\mathcal{D}}'_q(p) = \max \mathcal{D}^p_{m,n} \sum_{p'=1}^{w_o} \left(h^{p'} \sum_{cell_{m,n} \in q} \frac{\mathcal{D}^{p-1+p'}_{m,n}}{\max \mathcal{D}^{p-1+p'}_{m,n}} \right), \tag{21}$$

where $h^{p'}$ is the weight of period p' with $0 < h^{p'} < 1$.

UAV Dispatchment. Given the predicted data volumes of requests in different squares, we now dispatch UAVs to their hovering regions. Note that the UAV in a single square can implement the requests in its nearby squares. If a few neighbor squares have a surge of data volumes of requests, there may be a UAV in each of the squares. This unfortunately may lead to the underutilization of UAV computing resources. To further avoid UAV gathering around a few squares and leaving some squares unattended, we adopt an incremental deployment algorithm. In other words, we dispatch a UAV to square $\arg\max_q \hat{\mathcal{D}}'_q(p)$. The requests are then assigned to each dispatched UAV greedily. Afterward, we update $\hat{\mathcal{D}}'_q(p)$ by setting it to the remaining unassigned data volumes. The preceding procedure continues until all data volumes of requests are assigned to UAVs. The detailed steps are shown in Algorithm 2, referred to as `OL_LEARN`.

ALGORITHM 2: `OL_LEARN`

```
Input: $G = (V, E)$, a set $R$ of user requests that arrive into the system dynamically, a set $\mathcal{U}$ of UAVs.
Output: The total flow time of all requests in $R$.
1: Divide the hovering region into $M \times N$ cells with equal sizes, where each square have multiple cells;
2: for each period $p$ do
3:     Convert the data volumes of the cells to grayscale images;
4:     Predict the grayscale images of future $w_o$ periods, through adopting the prediction method in Figure 5, to obtain
       the predicted data volume $\hat{\mathcal{D}}'_q(p)$ of each square $q$;
5:     for each UAV $u$ in $\mathcal{U}$ do
6:         Dispatch the UAV to the square with the maximum remaining predicted data amount $\hat{\mathcal{D}}'_q(p)$;
7:         Update the $\hat{\mathcal{D}}'_q(p)$ as the remaining unprocessed data volume;
8:     Invoke algorithm OL to schedule requests in period $p$.
9: return Scheduling for all requests and the total flow time.
```

5.3 Algorithm Analysis

We now analyze the performance of the proposed algorithm `OL` in the following lemma and theorem. We first show that the obtained solution is feasible to the flow-time minimization problem by the following lemma.

Lemma 1. *The solution obtained by algorithm* `OL` *is feasible solution to the flow-time minimization problem.*

Please see the proof in the appendix.

Next, we analyze the competitive ratio of the proposed algorithm in the following theorem.

Theorem 1. *Given a UAV-aided MEC $G = (V, E)$ providing AI services for a set R of user requests, there exists an online algorithm (i.e., Algorithm 1) for the online flow-time minimization problem in a UAV-aided network with a given number of dispatched UAVs and given data volumes with a competitive ratio $O(\frac{k}{\rho+\epsilon})$, where $\rho = \max\{\rho_{j',t'}\}$ and ϵ is a constant.*

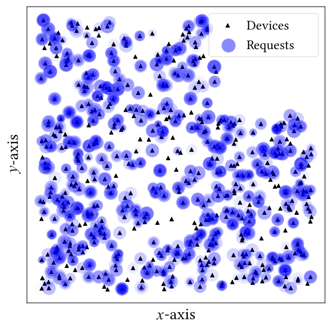
The proof of Theorem 1 can be found in Appendix A.2.

6 EXPERIMENTS

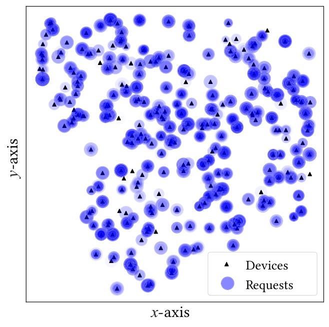
In this section, we evaluate the performance of the proposed algorithms by simulations using real data.

6.1 Parameter Settings

We considered a 1,000 × 1,000 m square area with two macro base stations, which is evenly divided into 25 squares. We used two different datasets to simulate the distribution of mobile users. One is



(a) The request heat map on the IoT dataset



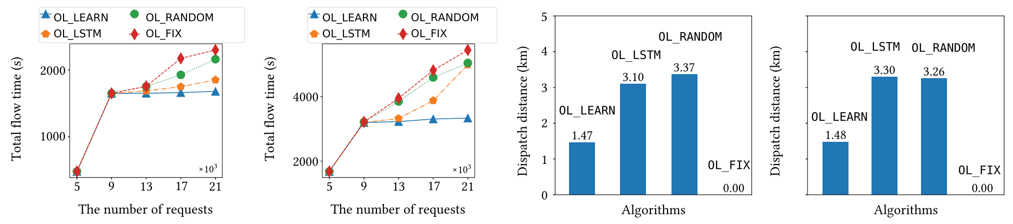
(b) The request heat map on the EUA dataset

Fig. 6. The request heat map, which shows the locations of requests in a geographical location.

the real-world location information of 6,080 mobile users in the dataset [29], and the other dataset has 272 mobile users in the EUA dataset [20]. Each mobile user generates requests following the Poisson distribution that is denoted by $P(\theta)$, where θ is the expected arrival rate of requests [29]. Furthermore, to simulate user requests under emergency events, we generated some emergency events and adjusted the expected arrival rate θ of the Poisson distribution when emergency events occurred, which is based on the dataset in other works [28, 56]. The request heat map in Figure 6 shows the location and request distribution of mobile users in a target area, where the black triangles represent mobile users and the blue circles represent requests. In addition, to further demonstrate the characteristics of requests, we used the size of circles to display data volumes of requests and the color shade of circles to distinguish arrival times of requests. Each UAV has an energy capacity of 3×10^5 J, and the energy consumption rate ζ on hovering is 150 J/s [21]. The communication distance of each UAV is 224 m, and the hovering height of a UAV is lower than its maximum communication distance. The data rate of wireless channel is set according to Zhang et al. [57]. The maximum power of all computing units of UAV P^{max} is 75 W, and the sum of P^{idle} and P^{leak} is 10 W [15, 26]. The transmitting power of mobile users and UAVs is 3.16 W [50]. The data volume D_j of each request is randomly withdrawn from $[0.5, 2]$ MB [25]. The duration of a time slot τ is set as 0.1 seconds, and there are 1,000 time slots in each time period p. The number w_o of looking-ahead periods of the prediction algorithm is 6 (i.e., 10 minutes). D_{unit} is set to 1 KB, and the delay δ of processing the data of each request is 0.2 seconds [43]. We set $k = 2$ of the l_k-norm.

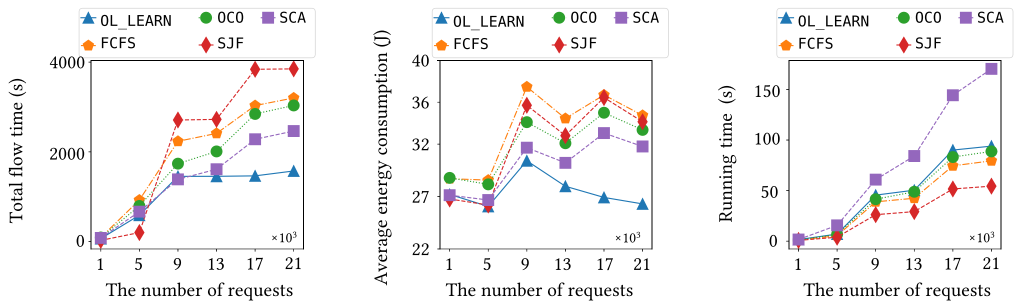
We compared the proposed algorithms with the following algorithms: (1) algorithm `OL` that adopts a one-time dispatchment of UAVs without redispatchment, referred to as `OL_FIX`; (2) algorithm `OL` that randomly dispatches UAVs, referred to as `OL_RANDOM`; (3) an algorithm that dispatches UAVs based on LSTM predictions and `OL`, referred to as `OL_LSTM`; (4) the first-come-first-served algorithm (i.e., `FCFS`), which schedules the earliest arrived request first; (5) the shortest job first, referred to as `SJF`, which schedules the request with the minimum data volume; (6) a successive convex approximation algorithm in the work of Yu et al. [49], referred to as `SCA`, which considers the optimization of task completion time in UAV-aided MEC; and (7) an online convex optimization method in the work of Liu et al. [27], referred to as `OCO`, which aims to strike a balance between fairness and latency. Since `FCFS`, `SJF`, `SCA`, and `OCO` do not consider the UAV dispatchment, we adopt the method in `OL_LEARN` for the UAV dispatchment.

Unless otherwise specified, these parameters will be adopted in the default setting. The values of the performance of the proposed algorithms are the average of 15 random draws of the default



(a) Total flow time on the IoT dataset (b) Total flow time on the EUA dataset (c) Average dispatch distance on the IoT dataset (d) Average dispatch distance on the EUA dataset

Fig. 7. The performance of algorithms OL_LEARN in terms of the total flow time and the dispatch distance of UAVs, by varying the number of requests.



(a) Total flow time (b) Average energy consumption (c) Running time

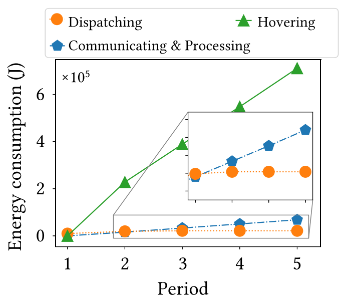
Fig. 8. The performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO in terms of the total flow time, the average energy consumption, and the running times, by varying the number of requests.

setting. The running times are obtained in a machine with a 3.20-GHz Intel Core i7-8700 CPU and 16 GiB of RAM.

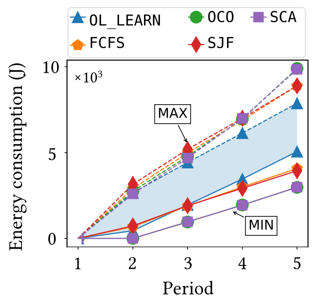
6.2 Performance Evaluation

First, we evaluated the performance of OL_LEARN against OL_FIX, OL_RANDOM, and OL_LSTM on both datasets with 10 UAVs. Figure 7(a) and (b) show that OL_LEARN has the lowest total flow time on both datasets. The reason is that OL_LEARN predicts the data volumes of requests via the spatial-temporal prediction of convLSTM, which can capture clustering and mobility of mobile users. Meanwhile, it adopts an incremental deployment strategy to optimize the coverage of base stations in each hovering region. It can be seen from Figure 7(c) and (d) that OL_LEARN reduces the average dispatch distance of UAVs in each period effectively, because a multi-step look-ahead prediction method is adopted.

Second, we studied the performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO on the IoT dataset with 10 UAVs by varying the number of requests from 1,000 to 21,000. It can be seen from Figure 8(a) that the total flow time by OL_LEARN is the lowest, due to the following two reasons. First, OL guarantees the fairness on scheduling requests via the RR policy. Second, OL_LEARN adopts a multi-timescale prediction method that accurately predicts the data volumes of requests, whereas the remaining algorithms adopt the expected data rates of requests. This avoids the cases when requests with bursty resource demands occupy all the resources, thereby causing the delay of implementing other requests. Since OL_LEARN leverages a tradeoff of the total flow time and energy consumption, in Figure 8(b) it is observed that OL_LEARN has the lowest average energy



(a) The total energy consumption of the dispatched 10 UAVs



(b) The maximum and minimum energy consumption of a UAV

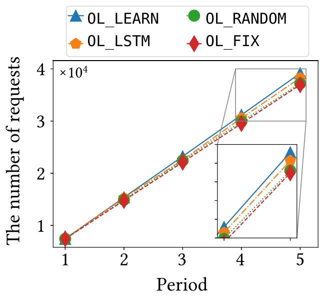
Fig. 9. The minimum and maximum energy consumptions of 10 UAVs due to communicating and processing with 40,000 requests in different periods.

consumption per request of the algorithms. In addition, from 9,000 to 21,000 requests, the average energy consumption of OL_LEARN keeps decreasing, whereas that of other algorithms fluctuates. In other words, algorithm OL can well control the average energy consumption as the number of requests increases. This is because OL assigns each request to an appropriate UAV, by considering the impact of the current scheduling request on the admissions of other requests and the RR scheduling policy. From Figure 8(c), we can see that the running time of OL_LEARN is at around 70 seconds, almost the same as OCO and FCFS.

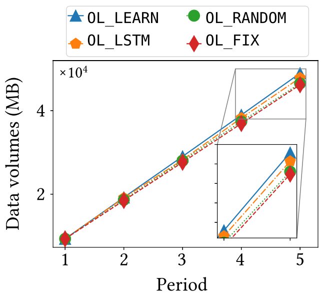
Third, we dealt with the total energy consumption of UAVs of algorithm OL_LEARN due to dispatching, hovering, and communicating and processing in different time periods p during the time when they are providing services for mobile users, by dispatching 10 UAVs to serve 40,000 requests. From Figure 9(a), we can see that the hovering energy accounts for the highest proportion of the total energy consumption of UAVs. To better demonstrate the gap in energy consumption between communicating and processing versus dispatching, we have partially enlarged the results in Figure 9(a). From the enlarged view, we can see that the energy consumption due to communicating and processing grows faster than that of dispatching. The reason is that we consider continuous processing of data streams from users, which usually lasts longer than the time spent on dispatching.

Fourth, we evaluated the minimum and maximum energy consumptions due to communicating and processing of the 10 UAVs of algorithms OL_LEARN, OCO, SCA, FCFS, and SJF with 40,000 requests. In Figure 9(b), each curve represents the energy consumption due to communicating and processing of an individual UAV, where a solid line shows the minimum energy consumption due to communicating and processing of a UAV and a dashed line represents the maximum one. We can see that the difference (shaded area with blue color) between the maximum and minimum energy consumptions of OL_LEARN is the smallest one due to communicating and processing of a UAV. Namely, OL_LEARN balances the energy consumption of UAVs, because it assigns requests as evenly as possible to each UAV to prevent some UAVs from running out of energy.

Finally, we evaluated the performance of OL_LEARN, OL_LSTM, OL_RANDOM, and OL_FIX in terms of the number of implemented requests and data volumes processed by 10 UAVs. As shown in Figure 10, algorithms OL_LEARN and OL_LSTM can enable UAVs to process more requests and data volumes. Besides, OL_LEARN outperforms OL_LSTM in terms of both the number of implemented requests and the data volume, because OL_LEARN fully utilizes the spatial-temporal information and side information of data volumes.

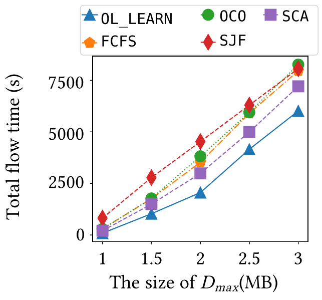


(a) The number of requests

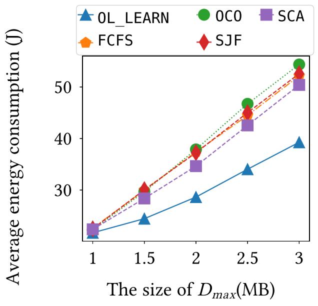


(b) Data volumes

Fig. 10. The number of requests and data volumes processed by 10 UAVs in algorithms OL_LEARN, OL_LSTM, OL_RANDOM, and OL_FIX in different periods.



(a) Total flow time



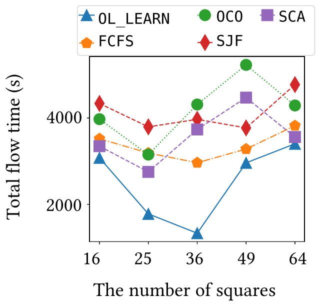
(b) Average energy consumption

Fig. 11. The impact of D_{max} ($Q = 25$, $|\mathcal{U}| = 10$) on the total flow time and the average energy consumption of UAVs.

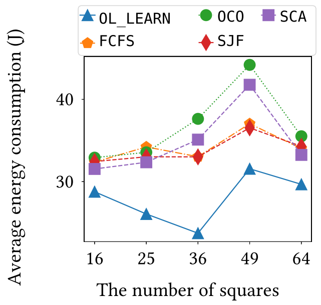
6.3 Impact of Parameters

We studied the impact of the maximum data volume D_{max} of requests on the performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO on the IoT dataset by varying D_{max} from 1 to 3 MB. From Figure 11, we can see that the total flow time obtained by all comparison algorithms increases with the growth of the maximum data volume. The reason is that larger data volumes usually mean longer transmission and processing times. As such, the average energy consumption of each request increases as well. The gap of the average energy consumption between OL_LEARN and the remaining algorithms increases with the growth of D_{max}. The results reveal the advantages of OL_LEARN in minimizing the total flow time of all requests.

We then evaluated the impact of the number Q of squares on the performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO on the IoT dataset by varying Q from 16 to 64. We can see from Figure 12(a) that the total flow time of OL_LEARN, FCFS, SJF, SCA, and OCO first decreases with the growth of Q, because more requests will have higher probabilities of being implemented in closer UAVs with low workloads. The reason is that a smaller value of Q usually means a larger area of each square. This leads to the uneven distribution of mobile users. In other words, when the size of each square is too large, some UAVs may hover in squares with few requests—that is, the number of requests admitted increases when the size of each square becomes smaller. However, as the number of squares grows from $Q = 36$, the flow time increases, because the growth of Q means that the area size of each square shrinks. As such, a number of UAVs gather around a

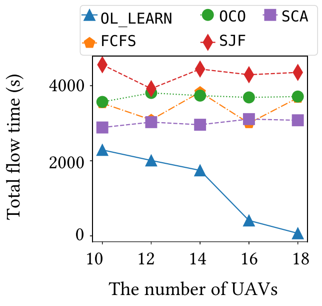


(a) Total flow time

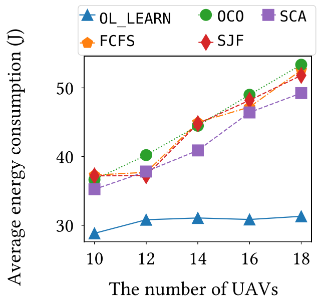


(b) Average energy consumption

Fig. 12. The impact of the number Q of squares ($D_j \in [5, 20]MB, |\mathcal{U}| = 9$) on the total flow time and the average energy consumption of UAVs.



(a) Total flow time



(b) Average energy consumption

Fig. 13. The impact of the number $|\mathcal{U}|$ of UAVs ($Q = 25, D_j \in [5, 20]MB$) on the total flow time and the average energy consumption of UAVs.

few squares with more requests, thereby making requests in other areas being implemented via longer transmission paths and increasing their flow times. In addition, as shown in Figure 12(b), the average energy consumption of the comparison algorithms decreases (or remains stable) when the number of squares increases from 16 to 36, and increases when Q increases. This is because as the number of squares increases, requests in the same area may be responded by more UAVs, meaning that each request can be implemented by energy-efficient UAVs. However, as Q keeps increasing, UAVs cluster in several squares with higher number of requests, making other requests being implemented with higher energy consumption.

We finally investigated the impact of the number $|\mathcal{U}|$ of UAVs on the performance of OL_LEARN, FCFS, SJF, SCA, and OCO on the IoT dataset by varying $|\mathcal{U}|$ from 10 to 18. From Figure 13, we can see that the total flow time of OL_LEARN keeps decreasing, and the flow time of FCFS, SJF, SCA, and OCO fluctuates. This is due to the fact that the system will have more processing capacity with the growth of $|\mathcal{U}|$; however, FCFS, SJF, SCA, and OCO do not give equal sharing among UAVs to all requests. In addition, we can see that the average energy consumption of the algorithms is generally on the rise with the growth of $|\mathcal{U}|$. The reason is that the system will consume more energy on hovering with the growth of $|\mathcal{U}|$.

6.4 Discussion on Implementation in the Real World

The proposed algorithms can be applied to real-world applications. We now discuss their computational costs and reliability issues:

— *Computational cost*: The key time-consuming parts of the proposed algorithm `OL` is the RR policy in each time slot. It must be mentioned that the RR policy is a widely adopted scheduling methods in many operating systems. Basic RR policies have been proven to be efficient in real scenarios. Considering that the proposed RR policy includes an additional weight function and the function can be calculated efficiently, the proposed `OL` algorithm can be applied in real scenarios. Besides, algorithm `OL_LEARN` has a convLSTM-based prediction method based on gray images. Given that similar methods are applied in real-time language processing, they can be efficient in the data volume prediction as well.

— *Reliability*: Although the proposed algorithms do not consider the reliability of UAVs, the proposed methods can be easily applied to consider the reliability requirements in real scenarios. For instance, we could set a candidate set of UAVs that are reliable to provide implement requests in the beginning of each time slot. In addition, when a UAV is out of service while requests are being processed, we could set checking points and resume the failed requests, as the requests are scheduled following an RR policy.

7 CONCLUSION AND FUTURE WORKS

In this article, we studied the problem of joint service caching and task offloading for the processing of data streams of user requests in UAV-aided MEC. We aim to minimize the total flow time of all user requests while meeting both computing and energy resource capacity constraints on UAVs. We proposed a learning optimization framework, and an online algorithm with a competitive ratio for the problem, by leveraging the RR scheduling and dual fitting analysis techniques. We evaluated the performance of the proposed online algorithm against existing studies empirically. Experimental results showed that the proposed online algorithms outperform existing studies by reducing the flow time at least 19% on average.

Future works of this study include the following. First, we plan to develop a hierarchy task offloading framework in multi-tier MEC networks for mobile users with various requirements. Specifically, besides offloading delay-sensitive user requests to small-cell base stations, resource-hungry user requests can be offloaded to macro base stations or even remote data centers with abundant computing resource. Second, we will investigate the heterogeneous capabilities of UAVs. Dispatching such heterogeneous UAVs to squares in the MEC plays a vital role in minimizing the flow time of mobile users. Third, we will extend the proposed algorithms to consider the stability of the system. Specifically, we may assume that each UAV has a probability of providing continuous stream processing in a number of future time slots. While offloading tasks to UAVs, we may only consider the UAVs that would guarantee the stability of executing tasks to a certain requirement (i.e., stable with 0.99 probability). The stability of task offloading in UAV networks, however, can be further explored to consider various network status and various types of tasks. Therefore, we will consider stable task offloading in our future work. Fourth, we also will consider the flow-time minimization problem with heterogeneous applications, and each application may have different requests. Considering that different applications may have complex influences on each other, we will design novel graph-based spatial-temporal prediction methods to predict the data volumes of different types of applications.

APPENDIX

A PROOF DETAILS

In this section, we provide proof details of Lemma 1 and Theorem 1. To this end, we first give the dual of LP in the following. Let α_j, β_t, and γ_j be the dual variables of Constraints (16), (17), and

(18), respectively. The dual objective then is to

$$\textbf{LP-DUAL:}\ \max \sum_j \alpha_j - \sum_t \beta_t - \sum_j \gamma_j, \tag{22}$$

subject to

$$\frac{\alpha_j}{D_j} - \frac{\beta_t}{|\mathcal{U}|LD_{max}} - \frac{\gamma_j L e'_{unit}}{EN} \le \left(\frac{(t-a_j)^k}{D_j} + \frac{D_j^k}{D_j}\right), \tag{23}$$

$$\alpha_j, \beta_t, \gamma_j \ge 0, \tag{24}$$

where k is a constant with $1 \le k \le 3$ in the l_k-norm. We now define a new setting for dual variables to capture the marginal impact of the arrival of request r_j. Motivated by Im et al. [17], we divide time slots into two subsets: the set T_o of *overloaded time slots* and the set T_u of *underloaded time slots*, respectively. In overloaded time slots in T_o, UAVs are busy in the RR's schedule as the number of requests assigned to each UAV is higher than the number L of its virtual UAVs. This means that there exists at least one virtual UAV having multiple requests scheduled to it. For the underloaded time slots in T_u, each virtual UAV has at most one request assigned to it.

The assignment of request r_j impacts the alive requests that are currently under executions in overloaded and underloaded time slots. We adopt an accounting method to capture this impact on the flow time of requests. Let $R_{alv}(t, \le a_j)$ be the set of alive requests arrived prior to r_j at time slot t. Specifically, we set α_j to

$$\alpha_j = \sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \le a_j)}} \left(\rho_{j',t'} \cdot k \cdot (t' - a_{j'})^{k-1}\right)/n_{t'} \tag{25}$$

$$+ \sum_{t' \in [a_j, C_j] \cap T_u; j' \in R_{alv}(t, \le a_j)} k \cdot (t' - a_{j'})^{k-1} \tag{26}$$

$$- \epsilon \cdot F_j^k, \tag{27}$$

where ϵ is a constant. The setting of parameter $\rho_{j',t'}$ in α_j is because r_j may only be assigned to portion of the UAVs, and it may not impact the requests arrived at time slot t' with $t' < t$. Request r_j can only be assigned to a virtual UAV if the UAV's transmission range covers the mobile user of request r_j. In Equation (26), each virtual UAV has at most one assigned request, and its impact on the total flow time is not amortized by other requests. Since $R_{alv}(t, \le a_j)$ contains request r_j, we have to subtract the flow time of r_j by $\epsilon \cdot F_j^k$.

We then set dual variables β_t and γ_j. The scheduling of r_j has an immediate impact on Constraint (17) during the time period when it is alive, and also has the following impact when it completes. Since $\beta_t = \sum_j \beta_{jt}$, $\beta_{jt} = (1/2 - 3\epsilon) \cdot 1(t \in [a_j, C_j + \lambda \cdot F_j]) \cdot F_j^{k-1}$, where λ denotes a parameter that captures the length of time of r_j's impact after its completion, and $1(t \in [a_j, C_j + \lambda \cdot F_j]) = 1$ if $t \in [a_j, C_j + \lambda \cdot F_j]$ and 0 otherwise. Similarly, we set the dual variable γ_j by

$$\gamma_j = \left(\sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \le a_j)}} \frac{\rho_{j',t'}(\frac{1}{4} - 2\epsilon)e'_{unit}}{n_{t'}} \cdot F_j^{k-1}\right)$$
$$+ \left(\sum_{\substack{t' \in [a_j, C_j] \cap T_u \\ j' \in R_{alv}(t, \le a_j)}} (1/4 - 2\epsilon)e'_{unit} F_j^{k-1}\right) - \epsilon e'_{unit} F_j^k.$$

Given the dual variable settings, our analysis follows three steps.

A.1 Proof of Lemma 1

Proof. We first show the dual feasibility by showing that the dual Constraint (23) is met.

We bound α_j/D_j as follows:

$$\frac{\alpha_j}{D_j} = \frac{1}{D_j}\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_o\\ j'\in R_{alv}(t,\le a_j)}} (\rho_{j',t'}k(t'-a_{j'})^{k-1})/n_{t'}\Big) \tag{28}$$

$$+\frac{1}{D_j}\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_u\\ j'\in R_{alv}(t,\le a_j)}} k(t'-a_{j'})^{k-1}\Big) - \frac{1}{D_j}\epsilon\cdot F_j^k. \tag{29}$$

We analyze the bounds of terms (28) and (29) of the right-hand side of the preceding equation, respectively. Specifically, assuming that $e'_{unit}\le 1/D_j$ and $k\cdot D_j\ge \epsilon F_j$, we have

$$(1/D_j)\Big(\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_u\\ j'\in R_{alv}(t,\le a_j)}} k\cdot(t'-a_{j'})^{k-1}\Big)-\epsilon\cdot F_j^k\Big) \tag{30}$$

$$\le (1/D_j)(D_j/\mu)k\cdot F_j^{k-1}-(1/D_j)\epsilon\cdot F_j^k \tag{31}$$

$$\le (1/D_j)\big(k(k/\epsilon)^{k-1}D_j^k\big)-(\epsilon F_j^k)/D_j \tag{32}$$

$$\le k(k/\epsilon)^{k-1}D_j^k-\epsilon\cdot e'_{unit}\cdot F_j^k,\ \text{since } e'_{unit}\le 1/D_j$$

$$\le k(k/\epsilon)^{k-1}D_j^k-\epsilon\cdot e'_{unit}\cdot F_j^k+(L\cdot e'_{unit}/E)\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_u\\ j'\in R_{alv}(t,\le a_j)}}(1/4-2\epsilon)e'_{unit}F_j^{k-1}\Big). \tag{33}$$

The derivation from Equation (30) to Equation (31) is due to the fact that at each underloaded time slot, each request is guaranteed with a speed of μ to process its data. In addition, the derivation from Equation (31) to Equation (32) is due to the fact that $k\cdot D_j\ge\epsilon F_j$.

We then analyze the bound of (28). Note that the scheduling of request r_j needs to meet Constraint (17). r_j contributes to β_t before and after its completion. The reason is that the resource it occupied may influence the scheduling of other future requests, as the virtual UAVs are overloaded and may take much longer times to finish their assigned requests.

Let $B(t')$ be the set of requests that contribute to $\beta_{t'}$ by a positive quantity (i.e., $B(t')=\{r_{j'}\mid t'\in[a_{j'},C_{j'}+\lambda F_{j'}]\}$). We thus divide Equation (28) into two parts—that is,

$$(28) = 1/D_j\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_o\\ j'\in R_{alv}(t,\le a_j)}} (\rho_{j',t'}\cdot k\cdot(t'-a_{j'})^{k-1})/n_{t'}\Big)$$

$$=\frac{1}{D_j}\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_o\\ j'\in R_{alv}(t,\le a_j)\backslash B(t)}} \frac{\rho_{j',t'}\cdot k\cdot(t'-a_{j'})^{k-1}}{n_{t'}}\Big) \tag{34}$$

$$+\frac{1}{D_j}\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_o\\ j'\in R_{alv}(t,\le a_j)\cap B(t)}} \frac{\rho_{j',t'}\cdot k\cdot(t'-a_{j'})^{k-1}}{n_{t'}}\Big). \tag{35}$$

We now analyze the bound of in Equation (34). In other words, for any request r_j and $t \geq a_j$, let $\rho = \max\{\rho_{j',t'}\}$, and we have

$$
\begin{aligned}
&(1/D_j) \sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \setminus B(t)}} (\rho_{j',t'} \cdot k \cdot (t' - a_{j'})^{k-1})/n_{t'} \\
&\leq (1/D_j) \sum_{\substack{t' \in [a_j, \min\{t, C_j\}] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \setminus B(t)}} (\rho \cdot k \cdot (t' - a_{j'})^{k-1})/n_{t'} \\
&\leq (1/D_j) \sum_{\substack{t' \in [a_j, \min\{t, C_j\}] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \setminus B(t)}} \Big((\rho \cdot k \cdot (t' - a_{j'})^{k-1})/n_{t'} + \big((1/2)\rho(1/4 - 2\epsilon)e'_{unit}F_{j'}^{k-1}\big)/n_{t'}\Big) \\
&\leq \frac{1}{D_j} \sum_{\substack{t' \in [a_j, \min\{t, C_j\}] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \setminus B(t)}} \Big(\frac{\rho \cdot k \cdot (\frac{1}{\lambda})^{k-1} \cdot (t' - a_{j'})^{k-1}}{n_{t'}} \\
&\qquad + \big((1/2)\rho(1/4 - 2\epsilon)e'_{unit}F_{j'}^{k-1}\big)/n_{t'}\Big), \text{since } \frac{1}{\lambda} \geq 1 \\
&\leq (1/D_j)k \cdot (1/\lambda)^{k-1} \cdot (t' - a_{j'})^{k-1} \\
&\qquad + \frac{L \cdot e'_{unit}}{2E} \sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \setminus B(t)}} \Big(\frac{\rho(1/4 - 2\epsilon)e'_{unit}F_{j'}^{k-1}}{n_{t'}}\Big), \text{since } \frac{L \cdot e'_{unit}}{EN} > 1 \text{ and } D_j \geq 1. \qquad (36)
\end{aligned}
$$

The bound of Equation (35) can be derived similarly, omitted due to space limitations. In other words, assuming that $\mu L \leq (1/4 - 2\epsilon)e'_{unit}F_j^{k-1}$ for each request r_j and $t \geq a_j$, we have

$$
\begin{aligned}
&(1/D_j) \sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \cap B(t)}} (\rho_{j',t'} \cdot k \cdot (t' - a_{j'})^{k-1})/n_{t'} \\
&\leq \beta_t/(Q \cdot L \cdot D_{max}) + (L \cdot e'_{unit})/(2 \cdot E) \sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t, \leq a_j) \cap B(t)}} \Big(\frac{\rho(1/4 - 2\epsilon)e'_{unit}F_{j'}^{k-1}}{n_{t'}}\Big). \qquad (37)
\end{aligned}
$$

Combining inequalities (33), (36), and (37), we conclude that the dual constraint is met.

We then show the solution feasibility of the algorithm, which is to show that Constraints (8), (10), and (11) are met. We can show the feasibility according to the setting of virtual UAVs. □

A.2 Proof of Theorem 1

Proof. We first show the bound on the dual objective in Equation (22). To this end, we show the lower bound of $\sum_j \alpha_j$. Let $R_{alv}(t)$ be the set of alive requests in time slot t, then

$$\sum_j \alpha_j = \sum_j \Bigg(\Bigg(\sum_{\substack{t' \in [a_j, C_j] \cap T_o \\ j' \in R_{alv}(t', \le a_j)}} \frac{(\rho_{j',t'} \cdot k \cdot (t' - a_{j'})^{k-1})}{n_{t'}}\Bigg) + \Bigg(\sum_{\substack{t' \in [a_j, C_j] \cap T_u \\ j' \in R_{alv}(t', \le a_j)}} k \cdot (t' - a_{j'})^{k-1}\Bigg) - \epsilon F_j^k\Bigg) \tag{38}$$

$$= \Bigg(\sum_{\substack{t' \in T_o \\ j \in R_{alv}(t')}} \rho_{j,t'} \cdot k \cdot (t' - a_j)^{k-1} |R_{alv}(t', \ge a_j)| / n_{t'}\Bigg) + \Bigg(\sum_{t' \in T_u; j \in R_{alv}(t')} k \cdot (t' - a_j)^{k-1}\Bigg) - \epsilon \cdot RR \tag{39}$$

$$\ge 1/2 \sum_{t' \in T_o; j \in R_{alv}(t')} \rho/b \cdot k \cdot (t' - a_j)^{k-1} + \Bigg(\sum_{t' \in T_u; j \in R_{alv}(t')} k \cdot (t' - a_j)^{k-1}\Bigg) - \epsilon \cdot RR \tag{40}$$

$$\ge (\rho/2b - \epsilon) RR, \tag{41}$$

where $R_{alv}(t, \ge a_j)$ is the set of alive requests at time slot t and have arrived no earlier than r_j, RR is the objective of RR scheduling, $\rho/b = \min\{\rho_{j,t'}\}$, and b is a constant. The derivation from Equation (38) to Equation (39) is because request r_j is counted by every request that arrives after time slot a_j. To derive inequality (40), we see the requests in $R_{alv}(t)$ in pairs: the earliest arrived request is counted by $n_{t'}$ times while the latest arrived request is counted once. Then,

$$(\rho \cdot k \cdot (t' - a_j)^{k-1} \cdot n_{t'} + \rho \cdot k \cdot (t' - a_i)^{k-1})/n_{t'} \tag{42}$$
$$\ge (n_{t'} + 1)/2n_{t'}(\rho \cdot k \cdot (t' - a_j)^{k-1} + \rho \cdot k \cdot (t' - a_i)^{k-1}) \ge \rho/2(k \cdot (t' - a_j)^{k-1} + k \cdot (t' - a_i)^{k-1}).$$

Summing over all pairs in $R_{alv}(t)$, inequality (40) follows.

We then show an upper bound on $\sum_t \beta_t$. By the definition of β_t,

$$\sum_t \beta_t = \sum_{j,t} \beta_{jt} = \sum_j (1 + \lambda) F_j (1/2 - 3\epsilon) F_j^{k-1}$$
$$= (1 + \lambda)(1/2 - 3\epsilon) RR \le (1/2 - 3\epsilon + \rho/4) RR, \text{if } \lambda = \rho/2.$$

We finally show an upper bound on $\sum_j \gamma_j$—that is,

$$\sum_j \gamma_j = \Bigg(\sum_{\substack{t' \in T_o \\ j \in R_{alv}(t')}} \frac{\rho_{j,t'}(\frac{1}{4} - 2\epsilon) e'_{unit} |R_{alv}(t', \ge a_j)| F_j^{k-1}}{n_{t'}}\Bigg)$$
$$+ \Bigg(\sum_{\substack{t' \in T_o \\ j \in R_{alv}(t')}} (1/4 - 2\epsilon) e'_{unit} F_j^{k-1}\Bigg) - \epsilon e'_{unit} F_j^k \tag{43}$$

$$< (1/4 - 2\epsilon) e'_{unit}\Bigg(\sum_{t' \in T_o, j \in R_{alv}(t')} \rho F_j^{k-1} + \sum_{t' \in T_o, j \in R_{alv}(t')} F_j^{k-1}\Bigg) - \epsilon \cdot e'_{unit} \cdot RR \tag{44}$$

$$\le \big((1/2 - 4\epsilon) e'_{unit} - \epsilon e'_{unit}\big) RR, \text{ since } \rho \le 1 \tag{45}$$

$$\le (1/2 e'_{unit} - 5 e'_{unit} \epsilon) RR. \tag{46}$$

The derivation from (43) to (44) is similar to that in (42). In other words, considering two requests that are counted by $n_{t'}$ and $n_{t'} - 1$ times, respectively, we have $(\rho F_j^{k-1} \cdot n_{t'} + \rho F_i^{k-1} \cdot (n_{t'} - 1))/n_{t'} < \rho F_i^{k-1}$.

By the dual variable settings, the objective is $\Omega(\rho + \epsilon)$ times RR's kth power. Let OPT_{ILP} and OPT_{LP} be the optimal solutions of ILP and LP, respectively. Let F' be the obtained solution by algorithm OL. We have $F'/OPT_{LP} \ge c \cdot (\rho + \epsilon)$, where c is a given constant. By using the same argument as that of Im et al. [17], we have $OPT_{LP} \leq 2 \cdot OPT_{ILP}$. The competitive ratio of algorithm OL is thus $O(\frac{k}{\rho+\epsilon})$. □

REFERENCES

[1] AWS. 2023. 5G Edge Computing Infrastructure: AWS Wavelength. Retrieved March 1, 2023 from https://aws.amazon.com/wavelength

[2] AWS. 2023. 5G and Edge Computing Enhances Connected and Autonomous Experiences. Retrieved March 1, 2023 from https://d1.awsstatic.com/autonomous-connected.pdf

[3] Arman Azizi, Saeedeh Parsaeefard, Mohamad R. Javan, Nader Mokari, and Halim Yanikomeroglu. 2020. Profit maximization in 5G+ networks with heterogeneous aerial and ground base stations. *IEEE Transactions on Mobile Computing* 19, 10 (2020), 2445–2460.

[4] Nikhil Bansal and Kirk R. Pruhs. 2010. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM Journal on Computing* 39, 7 (2010), 3311–3335.

[5] Suzhi Bi, Liang Huang, and Ying-Jun Angela Zhang. 2020. Joint optimization of service caching placement and computation offloading in mobile edge computing systems. *IEEE Transactions on Wireless Communications* 19, 7 (2020), 4947–4963.

[6] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking* 24, 5 (2016), 2795–2808.

[7] Mengyu Chen, Weifa Liang, and Sajal K. Das. 2021. Data collection utility maximization in wireless sensor networks via efficient determination of UAV hovering locations. In *Proceeding of the International Conference on Pervasive Computing and Communications (PerCom'21)*. IEEE.

[8] Mengyu Chen, Weifa Liang, and Yuchen Li. 2020. Data collection maximization for UAV-enabled wireless sensor networks. In *Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN'20)*. IEEE.

[9] Mingzhe Chen, Mohammad Mozaffari, Walid Saad, Changchuan Yin, Mérouane Debbah, and Choong S. Hong. 2017. Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1046–1061.

[10] Weiwei Chen, Zhou Su, Qichao Xu, Tom H. Luan, and Ruidong Li. 2020. VFC-based cooperative UAV computation task offloading for post-disaster rescue. In *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE.

[11] Shuping Dang, Osama Amin, Basem Shihada, and Mohamed S. Alouini. 2020. What should 6G be? *Nature Electronics* 3, 1 (2020), 20–29.

[12] Yao Du, Kezhi Wang, Kun Yang, and Guopeng Zhang. 2018. Energy-efficient resource allocation in UAV based MEC system for IoT devices. In *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM'18)*. IEEE.

[13] Azade Fotouhi, Haoran Qiang, Ming Ding, Mahbub Hassan, Lorenzo G. Giordano, Adrian Garcia-Rodriguez, and Jinhong Yuan. 2019. Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3417–3442.

[14] Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu. 2019. Winning at the starting line: Joint network selection and service placement for mobile edge computing. In *Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM'19)*. IEEE.

[15] Sunpyo Hong and Hyesoon Kim. 2010. An integrated GPU power and performance model. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. ACM.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[17] Sungjin Im, Janardhan Kulkarni, and Benjamin Moseley. 2015. Temporal fairness of round robin: Competitive analysis of Lk-norms of flow time. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'15)*. ACM.

[18] Hyuk-Jin Jeong, Hyeon-Jae Lee, Chang H. Shin, and Soo-Mook Moon. 2018. IONN: Incremental offloading of neural network computations from mobile devices to edge servers. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC'18)*. ACM.

[19] Xiangqi Kong, Ning Lu, and Bin Li. 2021. Optimal scheduling for unmanned aerial vehicle networks with flow-level dynamics. *IEEE Transactions on Mobile Computing* 20, 3 (2021), 1186–1197.

[20] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang. 2018. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *Service-Oriented Computing*. Lecture Notes in Computer Science, Vol. 11236. Springer, 230–245.

[21] Yuchen Li, Weifa Liang, Wenzheng Xu, and Xiaohua Jia. 2020. Data collection of IoT devices using an energy-constrained UAV. In *Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS'20)*. IEEE.

[22] Yuchen Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, Xiaohua Jia, Yinlong Xu, and Haibin Kan. 2021. Data collection maximization in IoT-sensor networks via an energy-constrained UAV. *IEEE Transactions on Mobile Computing* 22, 1 (2021), 159–174.

[23] Weiwei Lin, Tiansheng Huang, Xin Li, Fang Shi, Xiumin Wang, and Ching-Hsien Hsu. 2021. Energy-efficient computation offloading for UAV-assisted MEC: A two-stage optimization scheme. *ACM Transactions on Internet Technology* 22, 1 (2021), 1533–5399.

[24] Chen-Feng Liu, Mehdi Bennis, Merouane Debbah, and H. Vincent Poor. 2019. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Transactions on Communications* 67, 6 (2019), 4132–4150.

[25] Chuanwen Luo, Meghana N. Satpute, Deying Li, Yongcai Wang, Wenping Chen, and Weili Wu. 2021. Fine-grained trajectory optimization of multiple UAVs for efficient data gathering from WSNs. *IEEE/ACM Transactions on Networking* 29, 1 (2021), 162–175.

[26] Cheng Luo and Reiji Suda. 2011. A performance and energy consumption analytical model for GPU. In *Proceedings of the 2011 IEEE 9th International Conference on Dependable, Autonomic, and Secure Computing (DASC'11)*. IEEE.

[27] Yang Liu, Huanle Xu, and Wing C. Lau. 2019. Online job scheduling with resource packing on a cluster of heterogeneous servers. In *Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM'19)*. IEEE.

[28] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. 2017. Time-series extreme event forecasting with neural networks at Uber. In *Proceedings of the Time Series Workshop (ICML'17)*.

[29] Claudio Marche, Luigi Atzori, Virginia Pilloni, and Michele Nitti. 2020. How to exploit the Social Internet of Things: Query generation model and device profiles' dataset. *Computer Networks* 174 (2020), 107248.

[30] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Merouane Debbah. 2015. Drone small cells in the clouds: Design, deployment and performance analysis. In *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM'15)*. IEEE.

[31] Yahoo! Finance. 2016. F-Cell technology from Nokia Bell Labs revolutionizes small cell deployment by cutting wires, costs and time. *GlobalNewswire*. Retrieved February 10, 2024 from https://finance.yahoo.com/news/f-cell-technology-nokia-bell-112005468.html

[32] Qualcomm. 2016. Paving the path to 5G: Optimizing commercial LTE networks for drone communication. Retrieved December 1, 2022 from https://www.qualcomm.com/news/onq/2016/09/06/paving-path-5g-optimizing-commercial-lte-networks-drone-communication

[33] Yuben Qu, Haipeng Dai, Haichao Wang, Chao Dong, Fan Wu, Song Guo, and Qihui Wu. 2021. Service provisioning for UAV-enabled mobile edge computing. *IEEE Journal on Selected Areas in Communications* 39, 11 (2021), 3287–3305.

[34] Zhen Qin, Hai Wang, Zhenhua Wei, Yuben Qu, Fei Xiong, Haipeng Dai, and Tao Wu. 2021. Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities. *IEEE Internet of Things Journal* 8, 24 (2021), 17290–17307.

[35] Walid Saad, Mehdi Bennis, and Mingzhe Chen. 2020. A vision of 6G wireless systems: Applications, trends, technologies, and open research problems. *IEEE Network* 34, 3 (2020), 134–142.

[36] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, Vol. 1. 802–810.

[37] Hanine Tout, Azzam Mourad, Nadjia Kara, and Chamseddine Talhi. 2021. Multi-persona mobility: Joint cost-effective and resource-aware mobile-edge computation offloading. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1408–1421.

[38] Global Times. 2021. China's High-Tech Companies Aid Henan Flood Rescues with Drones and Satellites. Retrieved December 1, 2022 from https://www.globaltimes.cn/page/202107/1229309.shtml

[39] Rui Wang, Yong Cao, Adeeb Noor, Thamer A. Alamoudi, and Redhwan Nour. 2020. Agent-enabled task offloading in UAV-aided mobile edge computing. *Computer Communications* 149, 324–331.

[40] Dawei Wei, Jianfeng Ma, Linbo Luo, Yunbo Wang, Lei He, and Xinghua Li. 2021. Computation offloading over multi-UAV MEC network: A distributed deep reinforcement learning approach. *Computer Networks* 199 (2021), 108439.

[41] Di Wu, He Xu, Zhongkai Jiang, Weiren Yu, Xuetao Wei, and Jiwu Lu. 2021. EdgeLSTM: Towards deep and sequential edge computing for IoT applications. *IEEE/ACM Transactions on Networking* 29, 4 (2021), 1895–1908.

[42] Qiufen Xia, Weifa Liang, and Wenzheng Xu. 2013. Throughput maximization for online request admissions in mobile cloudlets. In *Proceedings of the 38th Annual IEEE Conference on Local Computer Networks (LCN'13)*. 589–596.

[43] Qiufen Xia, Zheng Lou, Wenzheng Xu, and Zichuan Xu. 2020. Near-optimal and learning-driven task offloading in a 5G multi-cell mobile edge cloud. *Computer Networks* 176 (2020), 107276.

[44] Wenzheng Xu, Tao Xiao, Junqi Zhang, Weifa Liang, Zichuan Xu, Xuxun Liu, Xiaohua Jia, and Sajal K. Das. 2022. Minimizing the deployment cost of UAVs for delay-sensitive data collection in IoT networks. *IEEE/ACM Transactions on Networking* 20, 2 (2022), 812–825.

[45] Zichuan Xu, Lizhen Zhou, Sid Chi-Kin Chau, Weifa Liang, Qiufen Xia, and Pan Zhou. 2020. Collaborate or separate? Distributed service caching in mobile edge clouds. In *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE.

[46] Yu Xu, Tiankui Zhang, Yuanwei Liu, Dingcheng Yang, Lin Xiao, and Meixia Tao. 2021. UAV-assisted MEC networks with aerial and ground cooperation. *IEEE Transactions on Wireless Communications* 20, 12 (2021), 7712–7727.

[47] Yu Xu, Tiankui Zhang, Jonathan Loo, Dingcheng Yang, and Lin Xiao. 2021. Completion time minimization for UAV-assisted mobile-edge computing systems. *IEEE Transactions on Vehicular Technology* 70, 11 (2021), 12253–12259.

[48] Qinglin Yang, Xiaofei Luo, Peng Li, Toshiaki Miyazaki, and Xiaoyan Wang. 2019. Computation offloading for fast CNN inference in edge computing. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS'19)*. 101–106.

[49] Zhe Yu, Yanmin Gong, Shimin Gong, and Yuanxiong Guo. 2020. Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet of Things Journal* 7, 4 (2020), 3147–3159.

[50] Lei Yang, Haipeng Yao, Jingjing Wang, Chunxiao Jiang, Abderrahim Benslimane, and Yunjie Liu. 2020. Multi-UAV enabled load-balance mobile edge computing for IoT networks. *IEEE Internet of Things Journal* 7, 8 (2020), 6898–6908.

[51] Changsheng You and Rui Zhang. 2020. Hybrid offline-online design for UAV-enabled data harvesting in probabilistic LoS channels. *IEEE Transactions on Wireless Communications* 19, 6 (2020), 3753–3768.

[52] Liang Zhang and Nirwan Ansari. 2020. Latency-aware IoT service provisioning in UAV-aided mobile edge computing networks. *IEEE Internet of Things Journal* 7, 10 (2020), 10573–10580.

[53] Qixun Zhang, Jingran Chen, Lei Ji, Zhiyong Feng, Zhu Han, and Zhiyong Chen. 2020. Response delay optimization in mobile edge computing enabled UAV swarm. *IEEE Transactions on Vehicular Technology* 69, 3 (2020), 3280–3295.

[54] Xijian Zhong, Yan Guo, Ning Li, and Yancheng Chen. 2020. Joint optimization of relay deployment, channel allocation, and relay assignment for UAVs-aided D2D networks. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 804–817.

[55] Cheng Zhan, Han Hu, Xiufeng Sui, Zhi Liu, and Dusit Niyato. 2020. Completion time and energy optimization in the UAV-enabled mobile-edge computing system. *IEEE Internet of Things Journal* 7, 8 (2020), 7808–7822.

[56] Lingxue Zhu and Nikolay Laptev. 2017. Deep and confident prediction for time series at Uber. In *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW'17)*. IEEE.

[57] Tiankui Zhang, Yu Xu, Jonathan Loo, Dingcheng Yang, and Lin Xiao. 2020. Joint computation and communication design for UAV-assisted mobile edge computing in IoT. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 5505–5516.

[58] Gongming Zhao, Hongli Xu, Yangming Zhao, Chunming Qiao, and Liusheng Huang. 2020. Offloading dependent tasks in mobile edge computing with service caching. In *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE.

[59] Lipeng Zhu, Jun Zhang, Zhenyu Xiao, Xianbin Cao, Xiang-Gen Xia, and Robert Schober. 2020. Millimeter-wave full-duplex UAV relay: Joint positioning, beamforming, and power control. *IEEE Journal on Selected Areas in Communications* 38, 9 (2020), 2057–2073.

Received 26 December 2022; revised 17 November 2023; accepted 11 January 2024