Flow-Time Minimization for Timely Data Stream Processing in UAV-Aided MEC

***Abstract*—Unmanned Aerial Vehicles (UAVs) have gained increasing attentions by both academic and industrial communities, due to their flexible deployments and efficient line-of-sight communications. Recently, UAVs equipped with base stations have been envisioned as a key technology to provide ubiquitous 5G network services for mobile users. In this paper, we consider the timely processing of data streams from mobile users by network services in a UAV-aided Mobile Edge Computing (MEC) network, where each UAV carries a 5G small-cell base station for communication and data processing. 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 that is the duration from the issuing of a user request to its completion, subject to resource and energy capacities on UAVs. We then propose a spatial-temporal learning optimization framework. Built upon the framework, we devise an online algorithm with a competitive ratio for the problem, by leveraging the round-robin scheduling and dual fitting techniques. We finally evaluate the performance of the proposed algorithms. Experimental results show that the proposed algorithms outperform their comparison counterparts by reducing the flow time at least 19% on average.**

I. Introduction

Recently, the technique of UAVs carrying small-cell base stations with computing units (such as neural network accelerators and FPGAs) is considered as a novel mobile edge computing (MEC) technique in 5G and beyond 5G networks [7], [22], [26]. UAVs with small-cell base stations can serve mobile users in various emergent events [23], such as earthquakes and storm floods, by dynamically dispatching UAVs to extend the coverage areas of terrestrial macro base stations. For example, Wing Loong from China has been used to provide emergency communications in 2021 Henan flood [28]. Further, venues holding large-scale events, or traffic hotspots will have a surge in demand for network services, and the terrestrial macro base stations may be too overwhelmed and congested to serve such demand from mobile users. In such cases, UAVs can also implement computing intensive AI services by carrying AI accelerators and FPGAs, when terrestrial base stations are out of the range or congested in peak hours [5], [7], [26].

Data stream processing of mobile users in the emergent events is a fundamental type of network services in a UAV-aided MEC, such as continuous object recognition and video streaming processing. Such data streams usually require to be processed timely; otherwise, the emergent events may not be resolved. For example, if the data streams assigned to a congested macro base station cannot be alleviated timely by computing units in UAVs, users may have degraded quality of experiences (QoE). The *flow time*, which is defined as the time from the data generation time to its completion time of data processing, is emerging as an effective metric to guarantee the real-time data processing of UAVs. In this paper, we study the problem of flow-time minimization for processing data streams in a UAV-aided MEC, by jointly caching services from ground base stations and offloading user requests to the cached services in UAVs with small-cell base stations.

Minimizing the flow-time for user requests with continuous data streams in a UAV-aided MEC is challenging. First, the arrival times and the data volumes of user requests have spatial and temporal correlations and can be impacted by various side information, such as festivals, weather, etc. Thus, minimizing the flow time of user requests requires precise spatial-temporal predictions that embed such side information; otherwise, requests in some locations may experience prohibitively long processing delays. Second, the performance of service caching and task offloading depend 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 an other request with long processing times or latencies. How to 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 in a UAV-aided MEC, by joint service caching and task offloading for timely data stream processing in a UAV-aided MEC. Although there are studies on leveraging UAVs to collect data for Internet of Things (IoT) networks [1], [6], [12], [21], [38], [41], [42], the continuous data stream processing is largely ignored. Further, none of the studies aimed to minimize the flow time of requests. In addition, studies on service caching and task offloading have been explored in conventional MECs [11], [13], [31], [33], [45], instead of UAV-aided MECs.

The main contributions of this paper are as follows.

- We formulate a novel optimization problem of flow time minimization problem in a UAV-aided MEC, by formulating an Integer Linear Program (ILP) solution.
- We propose a spatial-temporal learning optimization framework, 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 remainder of the paper is organized as follows. Section II summarizes related studies. Section III introduces the

system model and defines the problem. Section IV presents an exact solution to the offline version of the problem. Section V details a learning-based optimization framework. Section VI evaluates the performance of the proposed algorithms, and Section VII concludes the paper.

II. Related Work

UAV-aided MECs are gaining much attention due to high flexibilities provided by UAVs. Most existing studies however focused on communications relay, content caching, and data collection or sensory coverage in UAV-aided networks [1], [6], [12], [16], [21], [38], [41], [42], [46]. Joint service caching and task offloading in UAV-aided MECs with flying base stations have not been considered in these mentioned studies. For example, Xu *et al.* [32] investigated the problem of minimizing the deployment cost of UAVs to collect data from IoT networks. Kong *et al.* [12] 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.* [41] 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.

None of the above mentioned studies can be directly or indirectly applied to data stream processing, as the problems are fundamentally different. For example, if UAVs serve as a communication relay, the computing resource allocation usually do not need to considered. Also, the data collection in UAV-enabled IoT networks assumes that UAVs only need to collect data without processing the collected data.

Meanwhile, there are several studies on provisioning data processing services in UAV-aided MECs [3], [4], [6], [15], [25], [29], [30], [34], [36], [37], [39], [44], they did not consider the timely processing of data streams from mobile users. Also, they did not find a non-trivial trade-off between the fairness on different user request processing and the responsiveness of network services. For example, Chen *et al.* [3] investigated a problem of optimizing the quality-of-experience of wireless devices in UAV networks. Yu *et al.* [36] 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.* [15] considered the coordination of individual rationality and social benefits in UAV-assisted MEC system. Although Zhang *et al.* [40] aimed to optimize the delay of processing data streams, but they did not consider the service caching and task offloading in MEC.

III. Preliminaries

A. System Model

We consider a UAV-aided MEC $G = (V, E)$, where a set of macro base stations and a set of UAVs jointly provide services for mobile users in a given area. 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 [5], [7], [26], as shown in Fig. 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. We have $V = \mathcal{BS} \cup \mathcal{U}$. The small-cell base station attached to each 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 represents the communication links between base stations, UAVs, and mobile users.

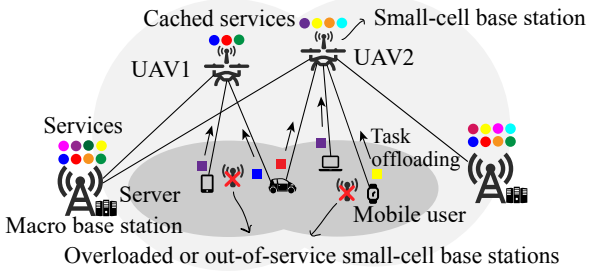


Fig. 1: An example of the UAV-aided MEC $G = (V, E)$.

B. Data Stream Processing, Aerial Service Caching and Task Offloading

AI services, such as video stream processing and real-time speech recognition, are deployed into macro base stations of the network. Such services require processing continuous data streams. When the macro base stations are congested, 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*. In this way, the UAVs with cached services then fly to the macro base stations to provide timely data stream processing. 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. Each r_j requires to stream an amount of data to a UAV for processing. Denote by D_j the total volume of data of r_j. Assuming that time is divided into equal time slots, and each time slot t lasts τ time units. Each r_j arrives into the system at a time slot that is denoted by a_j. Once r_j is scheduled, it can send its data stream to the MEC continuously until all its data D_j are uploaded for processing. Meantime, the system starts processing the uploaded data. Notice that requests arrive into the system dynamically. Also, the arrival time and data volumes of each request r_j are uncertain in the very beginning. Similar to [24], we assume that the processing result of each request is small in size, and the time of transmitting such results can be ignored.

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

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

Offloading r_j requires its service S_i being 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 $0 < \phi_j \leq 1$.

C. UAV Dispatching, Hovering, and Data Processing

In emergent events, 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 all 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 finished the data stream processing or its energy level is low. Unlike data collection in IoT, UAVs can finish the data collection of an IoT device quickly, where re-dispatching energy is the major energy consumption. Re-dispatchments of UAV thus are not considered in this paper.

There are infinite potential hovering locations for UAVs in the sky. Following standard practices [12], [16], 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 if each square has a single UAV, as shown in Fig. 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 $[1, Q]$, H_q is the hovering height of the UAV in square q.

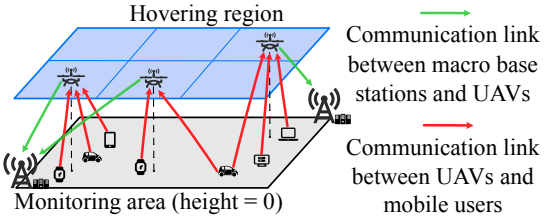


Fig. 2: The UAV hovering model.

Each dispatched UAV u can process the data as long as it receives the first unit of data of request r_j. 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 be 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 [19]. The data of each r_j can thus 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 $1 \leq z_{jtq} \leq D_j$.

D. Energy Models of UAVs

UAVs have limited battery power. Each dispatched UAV is recalled back for charge until it is done its service or using up 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 and returning for charge, which is denoted by EN^{other} and given as a priori. Let EN^{total} be the total energy capacity of a UAV. The energy consumption due to activities when hovering in its dispatched location can not be greater than $EN^{total} - EN^{other}$, which include, communicating with mobile users and macro base stations, processing offloading tasks, and hovering.

Each UAV $u \in \mathcal{U}$ receives data 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, $R_{j,q}$ is the achieved data rate of the wireless channel between the mobile user of r_j and the UAV in square q [44]. Recall that UAV u hovers at the center (x_q, y_q, H_q) of square q, 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 $R_{j,q}$.

Following existing studies [8], [18], 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 [34], [44], 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 [8], 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 on hovering. The energy consumption of each UAV on hovering thus is

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

E. 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. 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. We assume that 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 of its data [11], [35]. To speed up the processing, the UAV can assign more computing resources to process the data. Specifically, in time slot t with length τ, if UAV u wants to process z_{jtq} amount of data of r_j within $\tau - (D_{unit}/R_{j,q})$ time, the amounts of computing resource needed is

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

F. 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. This however may not be fair for some requests that are forced to wait for prohibitive 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

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

where k is a fixed integer in the range of $[1, 3]$. *The flow time minimization problem in a UAV-aided MEC* is to dispatch UAVs in $\mathcal{U}$ to the Q squares of a monitoring area to serve requests arrived within a given time horizon T (denote by R the set of all requests arrived for the 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.

IV. An ILP for the Offline Flow Time Minimization Problem

In each time slot t, a portion data of each request r_j may be uploaded 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 $x_{t,q}$ is 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

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

subject to the following constraints.

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

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

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

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

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

where $EN = EN^{total} - EN^{other}$. Constraint (8) says that request r_j can only be implemented after its arrival and all 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 the computing resource capacity of UAV u cannot be violated, where the LHS of Ineq. (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 longer time in time slot t, less time can be used for processing z_{jtq} amount of data. This means that more computing resource is needed to speed up the process to make sure the amount z_{jtq} of data is processed within time slot t. 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 $[1, D_j]$.

V. A Learning-Based Online Optimization Framework for the Flow Time Minimization Problem

We here propose a learning-based optimization framework for the problem. We first propose an online algorithm to schedule requests to the UAVs, if UAVs are dispatched and data volumes are given; otherwise, we devise a learning-based algorithm to dispatch UAVs.

A. An 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 requests to the dispatched UAVs. To this end, 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, and each virtual UAV u'_l is assigned to at most a single request. That is, 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. That is, 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, $R'_{\min} = \min\{R_{q,o}\}$ 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 that is scheduled for processing by one virtual UAV at time slot t, and $\mathbf{z'} = \{z'_{jt} \mid \forall t, r_j \in R\}$. Following studies in [2], [10], we approximate the objective in **ILP** by

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

subject to the following constraints.

$$\sum\nolimits_{t \geq a_j} z'_{jt} \geq D_j, \qquad \forall r_j \in R \quad (16)$$

$$\sum\nolimits_{j:t \geq a_j} z'_{jt} \leq |\mathcal{U}| \cdot L \cdot D_{max}, \quad \forall t \quad (17)$$

$$\sum\nolimits_{t} z'_{jt} \cdot e'_{unit} \cdot L \leq EN, \qquad \forall r_j \in R \quad (18)$$

$$0 \leq z'_{jt} \leq D_j, \quad (19)$$

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 the 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 virtual 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 we call the requests that are scheduled for execution and have not finished processing of all data as *alive requests*. To enhance the service responsiveness, we adopt an immediate-dispatch rule. That is, the assignment of a request has an impact on the admissions of other requests in time slot t. 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 tasks in the system, where the impact is defined as

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

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.

B. A Learning-Based Online Optimization Framework

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

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.

```
Create L virtual copies of each UAV u by Eq. (14);
for each time slot t do
    Assign each arrived request r_j to the central queue of the system;
    Sort unfinished and newly arrived requests in increasing order of their impact defined in Eq. (20), and denote by R_t sorted requests;
    for each request r_j in R_t do
        U'_j ← ∅ ;
        for each UAV u in U do
            if the mobile user of r_j is in the transmission range of UAV u then
                Add all virtual UAV u'_l of UAV u to set U'_j;
    Assign each request r_j in R_t to a virtual UAV in U'_j, by the RR scheduling policy;
    All assigned requests to each virtual UAV share the same amount of resource of the UAV;
return A scheduling for all requests.
```

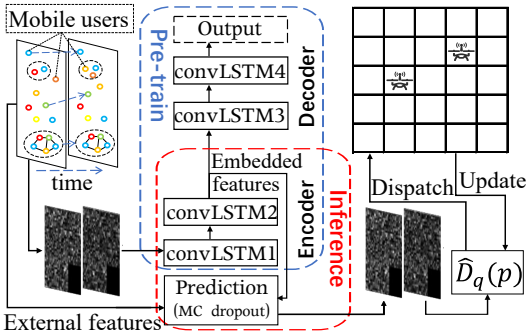


Fig. 3: The learning-based online optimization framework.

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 networks (CNN) architecture is a powerful tool to capture spatial information while the long short-term memory (LSTM) method [9] is capable to perform accurate time series predictions. We thus make a full customization of convLSTM [27] to predict the spatial and temporal patterns of data volumes of requests, by taking a series of grayscale images as inputs. The prediction results are finally used to guide the dispatchment of UAVs.

In the following, we describe the details of the proposed learning optimization framework.

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 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.

We now describe the prediction algorithm based on a series

of given grayscale images. Considering that the data volumes of requests are affected by many types of side information, such as festivals, weather, etc., 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 [43] by replacing the traditional LSTM with the convLSTM, as shown in Fig. 3. Specifically, the learning optimization framework first trains the encoder and decoder (blue dashed box in Fig. 3), using a series of grayscale images. Since the decoder cannot predict with external features, the framework then uses the trained encoder to train the neural network used for prediction (red dashed box in Fig. 3).

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 is that prediction each period may lead to frequent re-dispatchment of UAVs. Let w_o be the number of time periods that we look-ahead. However, considering 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,

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

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 neighbour 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 under utilization of UAV computing resources. To further avoid UAV gathering around a few squares and leave some squares unattended, we adopt an incremental deployment algorithm. That is, we dispatch a UAV to square $\arg\max_q \hat{\mathcal{D}}'_q(p)$. The requests are then assigned to each dispatched UAV greedily. Afterwards, we update $\hat{\mathcal{D}}'_q(p)$ by setting it to the remaining unassigned data volumes. The above 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.

C. Algorithm Analysis

We first give the dual of **LP**. 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

$$\textbf{LP-DUAL:} \ \max \sum\nolimits_j \alpha_j - \sum\nolimits_t \beta_t - \sum\nolimits_j \gamma_j, \quad (22)$$

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

$$\alpha_j, \beta_t, \gamma_j \geq 0. \quad (24)$$

We now define a new setting for dual variables to capture the marginal impact of the arrival of request r_j. Motivated by [10],

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.

```
Divide the hovering region into M × N cells with equal sizes, where each square have multiple cells;
for each period p do
    Convert the data volumes of the cells to grayscale images;
    Predict the grayscale images of future w_o periods, through adopting the prediction method in Fig. 3, to obtain the predicted data volume D̂'_q(p) of each square q;
    for each UAV u in U do
        Dispatch the UAV to the square with the maximum remaining predicted data amount D̂'_q(p);
        Update the D̂'_q(p) as the remaining unprocessed data volume;
    Invoke algorithm OL to schedule requests in period p.
return Scheduling for all requests and the total flow time.
```

we divide time slots into two sets: *overloaded time slots* and *underloaded time slots*, denote by T_o and T_u, 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 $R_{alv}(t, \leq a_j)$ be the set of alive requests arrived prior to r_j at time slot t. Specifically, we set α_j to

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

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

$$- \epsilon \cdot F_j^k. \quad (27)$$

The setting of parameter $\rho_{j',t'}$ in α_j is because that 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 Eq. (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 $R_{alv}(t, \leq a_j)$ 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 $\beta_t = \sum_j \beta_{jt}$, where $\beta_{jt} = (1/2 - 3\epsilon) \cdot 1(t \in [a_j, C_j + \lambda \cdot F_j]) \cdot F_j^{k-1}$, 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

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

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

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

Step 1: We first show the dual feasibility by showing that the dual Constraint (23) is met Lemma 1.

Lemma 1: The settings of the dual variables meet the dual Constraint (23) of **LP-DUAL**.

Proof: We bound α_j/D_j as follows.

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

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

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

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

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

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

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

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

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

The derivation from Ineq. (30) to (31) is due to that at each underloaded time slot, each request is guaranteed with a speed of μ to process its data. Also, the derivation from Ineq. (31) to (32) is due to 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 $B(t')$ by a positive quantity, i.e., $B(t') = \{r_{j'} \mid t'\in[a_{j'}, C_{j'}+\lambda F_{j'}]\}$. We thus divide Ineq. (28) into two parts, i.e.,

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

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

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

We now analyze the bound of Ineq. (34). That is, for any request r_j and $t\ge a_j$, let $\rho = \max\{\rho_{j',t'}\}$, we have

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

$$\le (1/D_j)\sum_{\substack{t'\in[a_j,\min\{t,C_j\}]\cap T_o \\ j'\in R_{alv}(t,\le a_j)\setminus B(t)}} (\rho\cdot k\cdot(t'-a_{j'})^{k-1})/n_{t'}$$

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

$$+\big((1/2)\rho(1/4-2\epsilon)e'_{unit}F_{j'}^{k-1}\big)/n_{t'}\Big)$$

$$\le \frac{1}{D_j}\sum_{\substack{t'\in[a_j,\min\{t,C_j\}]\cap T_o \\ j'\in R_{alv}(t,\le a_j)\setminus B(t)}} \Big(\frac{\rho\cdot k\cdot(\frac{1}{\lambda})^{k-1}\cdot(t'-a_{j'})^{k-1}}{n_{t'}}$$

$$+\big(1/2\rho(1/4-2\epsilon)e'_{unit}F_{j'}^{k-1}\big)/n_{t'}\Big), \text{since } 1/\lambda\ge 1$$

$$\le (1/D_j)k\cdot(1/\lambda)^{k-1}\cdot(t'-a_{j'})^{k-1} +$$

$$\frac{L\cdot e'_{unit}}{2E}\sum_{\substack{t'\in[a_j,C_j]\cap T_o \\ j'\in R_{alv}(t,\le a_j)\setminus B(t)}} \Big(\frac{\rho(1/4-2\epsilon)e'_{unit}F_{j'}^{k-1}}{n_{t'}}\Big),$$

since $L\cdot e'_{unit}/EN > 1$ and $D_j\ge 1$. (36)

The bound of Ineq. (35) can be derived similarly, omitted due to space limitation. That is, assuming that $\mu L \le (1/4-2\epsilon)e'_{unit}F_j^{k-1}$ for each request r_j and $t\ge a_j$, we have

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

$$\le \beta_t/(Q\cdot L\cdot D_{max}) + (L\cdot e'_{unit})/(2\cdot E)$$

$$\sum_{\substack{t'\in[a_j,C_j]\cap T_o \\ j'\in R_{alv}(t,\le a_j)\cap B(t)}} \Big(\frac{\rho(1/4-2\epsilon)e'_{unit}F_{j'}^{k-1}}{n_{t'}}\Big). \quad (37)$$

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

Step 2: We then show that the obtained solution is feasible to the flow time minimization problem by the following lemma.

Lemma 2: The solution obtained by algorithm OL is feasible.

Proof sketch: The solution feasibility of the algorithm is to show that Constraints (8), (10) and (11) are met. We can show the feasibility according to the setting of virtual UAVs. Omitted, due to space limitation.

Step 3: We finally analyze the competitive ratio.

Theorem 1: The competitive ratio of **Algorithm** 1 is $O(\frac{k}{\rho+\epsilon})$, where $\rho = \max\{\rho_{j',t'}\}$, and ϵ is a constant.

Proof: We first show the bound on the dual objective in Eq. (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,

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

$$+\Big(\sum_{\substack{t'\in[a_j,C_j]\cap T_u \\ j'\in R_{alv}(t',\le a_j)}} k\cdot(t'-a_{j'})^{k-1}\Big) - \epsilon\cdot F_j^k\Big) \quad (38)$$

$$= \big(\sum_{\substack{t'\in T_o \\ j\in R_{alv}(t')}} \rho_{j,t'}\cdot k\cdot(t'-a_{j'})^{k-1}|R_{alv}(t',\ge a_j)|/n_{t'}\big)$$

$$+\Big(\sum\nolimits_{t'\in T_o; j\in R_{alv}(t')} k\cdot(t'-a_{j'})^{k-1}\Big) - \epsilon\cdot RR \quad (39)$$

$$\ge 1/2\sum\nolimits_{t'\in T_o; j\in R_{alv}(t')} \rho/b\cdot k\cdot(t'-a_{j'})^{k-1}$$

$$+\Big(\sum\nolimits_{t'\in T_o; j\in R_{alv}(t')} k\cdot(t'-a_{j'})^{k-1}\Big) - \epsilon\cdot RR \quad (40)$$

$$\ge (\rho/2b-\epsilon)RR, \quad (41)$$

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 Eq. (38) to Eq. (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 by once; then,

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

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,

$$\sum\nolimits_t \beta_t = \sum\nolimits_{j,t} \beta_{jt} = \sum\nolimits_j (1+\lambda) F_j (1/2 - 3\epsilon) F_j^{k-1}$$
$$= (1+\lambda)(1/2 - 3\epsilon) RR \leq (1/2 - 3\epsilon + \rho/4) RR, \text{if } \lambda = \rho/2.$$

We finally show an upper bound on $\sum_j \gamma_j$, i.e.,

$$\sum_j \gamma_j = \Big(\sum_{\substack{t' \in T_o \\ j \in R_{alv}(t')}} \frac{\rho_{j,t'} (\frac{1}{4} - 2\epsilon) e'_{unit} | R_{alv}(t', \geq a_j) | F_j^{k-1}}{n_{t'}} \Big)$$
$$+ \Big(\sum\nolimits_{\substack{t' \in T_o \\ j \in R_{alv}(t')}} (1/4 - 2\epsilon) e'_{unit} F_j^{k-1} \Big) - \epsilon e'_{unit} F_j^k \quad (43)$$
$$< (1/4 - 2\epsilon) e'_{unit} \Big(\sum\nolimits_{t' \in T_o, j \in R_{alv}(t')} \rho F_j^{k-1}$$
$$+ \sum\nolimits_{t' \in T_o, j \in R_{alv}(t')} F_j^{k-1} \Big) - \epsilon \cdot e'_{unit} \cdot RR \quad (44)$$
$$\leq \big((1/2 - 4\epsilon) e'_{unit} - \epsilon e'_{unit} \big) RR, \quad \text{since } \rho \leq 1 \quad (45)$$
$$\leq (1/2 e'_{unit} - 5 e'_{unit} \epsilon) RR. \quad (46)$$

The derivation from (43) to (44) is similar to that in (42). That is, 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 $F'/OPT_{LP} \geq c \cdot (\rho + \epsilon)$, where c is a given constant. By using the same argument as that of [10], we have $OPT_{LP} \leq 2 \cdot OPT_{ILP}$. The competitive ratio of algorithm OL thus is $O(\frac{k}{\rho + \epsilon})$. ■

VI. Experiments

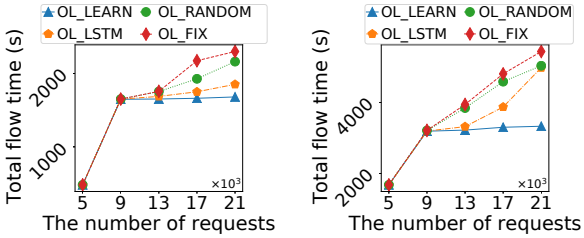
A. Parameter Settings

We consider $1,000m \times 1,000m$ square area with two macro base stations, which is evenly divided into 25 squares. We use two different datasets to simulate the distribution of mobile users. One is the real-world location information of 6,080 mobile users in the dataset [20] and the other dataset has 272 mobile users in the EUA-dataset [14]. Each mobile user generates requests following the Poisson distribution. Each UAV has an energy capacity of 3×10^5 $joules$, the energy consumption rate ζ on hovering is 150 J/s. The communication distance of each UAV is 224 meters, and the hovering height of a UAV is lower than its maximum communication distance. The data rate of wireless channel is set according to [44]. The maximum power of all computing units of UAV P^{max} is 75 W, and the sum of P^{idle} and P^{leak} is $10W$ [8], [18]. The transmitting power of mobile users and UAVs is 3.16 W [37]. The data volume D_j of each request is randomly withdrawn from $[0.5, 2]$ Mega Bytes (MB) [17]. 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 $1KB$, and the delay δ of processing the data of each request is 0.2 seconds [31]. We set $k = 2$ of the l_k-norm. 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 setting. The running times are obtained in a machine with a 3.20GHz Intel Core i7-8700 CPU and 16 GiB RAM.

We compared the proposed algorithms with the following algorithms: (1) algorithm OL that adopts a one-time dispatchment of UAVs without re-dispatchment, 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 [36], referred to as SCA, which considers the optimization of task completion time in UAV-aided MEC; and (7) an online convex optimization method in [19], 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.

B. Performance Evaluation

We first evaluate the performance of OL_LEARN against OL_FIX, OL_RANDOM and OL_LSTM on both datasets with 10 UAVs. Fig. 4 (a) and (b) show that OL_LEARN has the lowest total flow time on both datasets. The reason is that the OL_LEARN predicts the data volumes of requests via the spatio-temporal prediction of the convLSTM, which can well capture clustering and mobility of mobile users. Meanwhile, it adopts an incremental deployment strategy, which can better optimize the coverage of base stations in each hovering region.

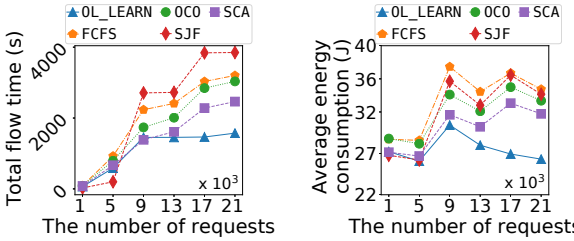


(a) Total flow time on IoT-dataset. (b) Total flow time on EUA-dataset.

Fig. 4: The performance of algorithms OL_LEARN.

We then study the performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO on IoT-dataset, with 10 UAVs by varying the number of requests from $1,000$ to $21,000$. We can see from Fig. 5 (a) that the total flow time by OL_LEARN is the lowest. The reason is that OL guarantees the fairness on scheduling requests via the RR policy. Since OL_LEARN leverages a tradeoff of the total flow time and energy consumption, in Fig. 5 (b) it is observed that OL_LEARN has the lowest

average energy consumption per request of the algorithms. We can also see that the average energy consumption of all algorithms increases first from $1,000$ to $5,000$ requests, peaking at $9,000$ requests, and decreases afterwards. This is because the amortized energy consumption decreases with the increase of the number of requests. However, as the number of requests keeps increasing, more UAVs are needed, leading to more energy consumption. The running time of OL_LEARN is at around 70 seconds almost the same as OCO and FCFS.



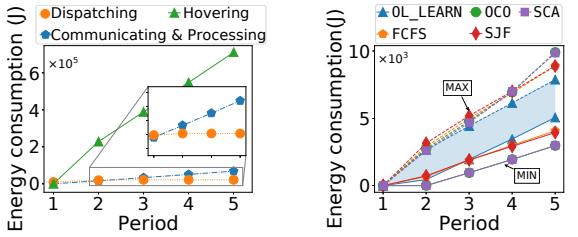
(a) Total flow time. (b) Average energy consumption.

Fig. 5: The performance of algorithms OL_LEARN, FCFS, SJF, SCA, and OCO.

We then investigate 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 a number of 40,000 requests. From Fig. 6 (a), we can see that the hovering energy accounts for the highest proportion of the total energy consumption of UAVs. In addition, the energy consumption due to communicating and processing grows the faster than dispatching. The reason is that we consider continuous processing of data streams from users, which usually lasts longer time than that spent on dispatching.

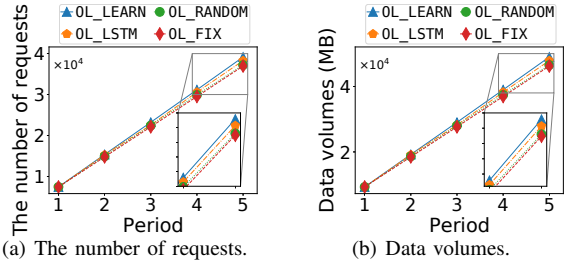
We also study the minimum and maximum energy consumption due to communicating and processing of the 10 UAVs of algorithms OL_LEARN, OCO, SCA, FCFS, and SJF with 40,000 requests. In Fig. 6 (b), each curve represents the energy consumption due to communicating and processing of an individual UAV, where 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 OL_LEARN has the smallest difference (shaded area with blue color) between the maximum and minimum energy consumption 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.

We also evaluate 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 Fig. 7, 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 spatio-temporal information and side information of data volumes.



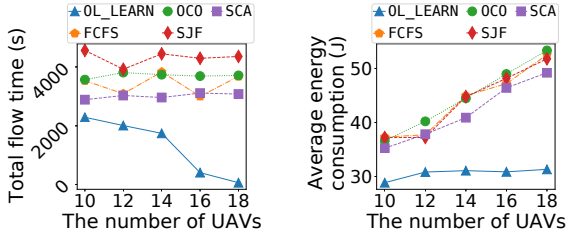
(a) The total energy consumption of the dispatched 10 UAVs. (b) The maximum and minimum energy consumption of a UAV.

Fig. 6: Energy consumption of UAVs.



(a) The number of requests. (b) Data volumes.

Fig. 7: The number of requests and data volumes processed by UAVs.



(a) Total flow time. (b) Average energy consumption.

Fig. 8: The impact of the number $|\mathcal{U}|$ of UAVs.

We finally evaluate the impact of the number $|\mathcal{U}|$ of UAVs on the performance of OL_LEARN, FCFS, SJF, SCA, and OCO on IoT-dataset, by varying $|\mathcal{U}|$ from 10 to 18. From Fig. 8, 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. Also, 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 $|\mathcal{U}|$.

VII. Conclusion

In this paper, we studied the problem of joint service caching and task offloading in a 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 round-robin scheduling and dual fitting analysis techniques. We evaluated the performance of the proposed online algorithm against existing studies empirically. Experimental results show that the proposed online algorithms outperform existing studies by reducing the flow time at least 19% on average.

REFERENCES

[1] A. Azizi, S. Parsaeefard, M. R. Javan, *et al.* Profit maximization in 5g+ networks with heterogeneous aerial and ground base stations. *IEEE Transactions on Mobile Computing*, Vol.19, No.10, pp. 2445–2460, 2020.

[2] N. Bansal and K. R. Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM Journal on Computing*, Vol. 39, No. 7, pp.3311–3335, SIAM, 2010.

[3] M. Chen, M. Mozaffari, W. Saad, *et al.* Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience. *IEEE Journal on Selected Areas in Communications*, Vol. 35, No. 5, pp. 1046–1061, 2017.

[4] W. Chen, Z. Su, Q. Xu, *et al.* VFC-based cooperative UAV computation task offloading for post-disaster rescue. *Proc. of INFOCOM*, IEEE, 2020.

[5] S. Dang, O. Amin, B. Shihada, *et al.* What should 6G be? *Nature Electronics*, Vol. 3, No. 1, pp. 20–29, Nature Publishing Group, 2020.

[6] Y. Du, K. Wang, K. Yang, *et al.* Energy-efficient resource allocation in UAV based MEC system for IoT devices. *Proc. of GLOBECOM*, IEEE, 2018.

[7] A. Fotouhi, H. Qiang, M. Ding, *et al.* Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges. *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 4, pp. 3417–3442, 2019.

[8] S. Hong and H. Kim. An integrated GPU power and performance model. *Proc. of ISCA*, ACM, 2010.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, MIT Press, 1997.

[10] S. Im, J. Kulkarni, and B. Moseley. Temporal fairness of round robin: Competitive analysis of Lk-norms of flow time. *Proc. of SPAA*, ACM, 2015.

[11] H. Jeong, H. Lee, C. Shin, and S. Moon. IONN: Incremental offloading of neural network computations from mobile devices to edge servers. *Proc. of SoCC*, ACM, 2018.

[12] X. Kong, N. Lu, and B. Li. Optimal scheduling for unmanned aerial vehicle networks with flow-level dynamics. *IEEE Transactions on Mobile Computing*, Vol.20, No. 3, pp. 1186–1197, 2021.

[13] C. Liu, M. Bennis, M. Debbah, and H. V. Poor. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Transactions on Communications*, Vol. 67, No. 6, pp. 4132–4150, 2019.

[14] P. Lai, Q. He, M. Abdelrazek, *et al.* Optimal edge user allocation in edge computing with variable sized vector bin packing. *Proc. of ICSOC*, Springer, 2018.

[15] W. Lin, T. Huang, X. Li, *et al.* Energy-efficient computation offloading for UAV-assisted MEC: a two-stage optimization scheme. *ACM Transactions on Internet Technology*, Vol. 22, No. 1, pp. 1533–5399, 2021.

[16] Y. Li, W. Liang, W. Xu, and X. Jia. Data collection of IoT devices using an energy-constrained UAV. *Proc. of IPDPS*, IEEE, 2020.

[17] C. Luo, M. N. Satpute, D. Li, *et al.* Fine-grained trajectory optimization of multiple UAVs for efficient data gathering from WSNs. *IEEE/ACM Transactions on Networking*, Vol. 29, No. 1, pp. 162–175, 2021.

[18] C. Luo and R. Suda. A performance and energy consumption analytical model for GPU. *Proc. of DASC*, IEEE, 2011.

[19] Y. Liu, H. Xu, and W. C. Lau. Online job scheduling with resource packing on a cluster of heterogeneous servers. *Proc. of INFOCOM*, IEEE, 2019.

[20] C. Marche, L. Atzori, V. Pilloni, and M. Nitti. How to exploit the social internet of things: Query generation model and device profiles dataset. *Computer Networks*, Vol. 174, p. 107248, Elsevier, 2020.

[21] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Drone small cells in the clouds: Design, deployment and performance analysis. *Proc. of GLOBECOM*, IEEE, 2015.

[22] Nokia. https://www.nokia.com/about-us/news/releases/2016/ 10/03/f-cell-technology-from-nokia-bell-labs-revolutionizes-small-cell-deployment-by-cutting-wires-costs-and-time. *Accessed on Apr. 2022*.

[23] Qualcomm. Paving the path to 5G: Optimizing commercial LTE networks for drone communication. *https://www.qualcomm.com/news/onq/2016/09/06/paving-path-5g-optimizing-commercial-lte-networks-drone-communication*. *Accessed on Apr. 2022*.

[24] Y. Qu, H. Dai, H. Wang, *et al.* Service provisioning for UAV-enabled mobile edge computing. *IEEE Journal on Selected Areas in Communications*, Vol. 39, No. 11, pp. 3287–3305, 2021.

[25] Z. Qin, H. Wang, Z. Wei, *et al.* Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities. *IEEE Internet of Things Journal*, Vol. 8, No. 24, pp. 17290–17307, 2021.

[26] W. Saad, M. Bennis, and M. Chen. A vision of 6G wireless systems: Applications, trends, technologies, and open research problems. *IEEE Network*, Vol. 34, No. 3, pp. 134–142, 2020.

[27] X. Shi, Z. Chen, H. Wang, *et al.* Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Proc. of NIPS*, Curran Associates, Inc., 2015.

[28] https://www.globaltimes.cn/page/202107/1229309.shtml. *Accessed on Apr. 2022*.

[29] R. Wang, Y. Cao, A. Noor, *et al.* Agent-enabled task offloading in UAV-aided mobile edge computing. *Computer Communications*, Vol. 149, pp. 324–331, Elsevier, 2020.

[30] D. Wei, J. Ma, L. Luo, *et al.* Computation offloading over multi-UAV MEC network: A distributed deep reinforcement learning approach. *Computer Networks*, Vol. 199, p. 108439, Elsevier, 2021.

[31] Q. Xia, Z. Lou, W. Xu, and Z. Xu. Near-optimal and learning-driven task offloading in a 5G multi-cell mobile edge cloud. *Computer Networks*, Vol. 176, p. 107276, Elsevier, 2020.

[32] W. Xu, T. Xiao, J. Zhang, *et al.* Minimizing the deployment cost of UAVs for delay-sensitive data collection in IoT networks. *IEEE/ACM Transactions on Networking*, Vol. 20, No. 2, pp. 812–825, 2022.

[33] Z. Xu, L. Zhou, S. Chi-Kin Chau, *et al.* Collaborate or separate? distributed service caching in mobile edge clouds. *Proc. of INFOCOM*, IEEE, 2020.

[34] Y. Xu, T. Zhang, Y. Liu, *et al.* UAV-assisted MEC networks with aerial and ground cooperation. *IEEE Transactions on Wireless Communications*, Vol. 20, No. 12, pp. 7712–7727, 2021.

[35] Q. Yang, X. Luo, P. Li, *et al.* Computation offloading for fast CNN inference in edge computing. *Proc. of RACS*, ACM, 2019.

[36] Z. Yu, Y. Gong, S. Gong, and Y. Guo. Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet of Things Journal*, Vol. 7, No. 4, pp. 3147–3159, 2020.

[37] L. Yang, H. Yao, J. Wang, *et al.* Multi-UAV enabled load-balance mobile edge computing for IoT networks. *IEEE Internet of Things Journal*, Vol. 7, No. 8, pp. 6898–6908, 2020.

[38] C. You and R. Zhang. Hybrid offline-online design for UAV-enabled data harvesting in probabilistic LoS channels. *IEEE Transactions on Wireless Communications*, Vol. 19, No. 6, pp. 3753–3768, 2020.

[39] L. Zhang and N. Ansari. Latency-aware IoT service provisioning in UAV-aided mobile edge computing networks. *IEEE Internet of Things Journal*, Vol. 7, No. 10, pp. 10573–10580, 2020.

[40] Q. Zhang, J. Chen, L. Ji, *et al.* Response delay optimization in mobile edge computing enabled UAV swarm. *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 3, pp. 3280–3295, 2020.

[41] X. Zhong, Y. Guo, N. Li, and Y. Chen. Joint optimization of relay deployment, channel allocation, and relay assignment for uavs-aided d2d networks. *IEEE/ACM Transactions on Networking*, Vol. 28, No. 2, pp. 804–817, 2020.

[42] C. Zhan, H. Hu, X. Sui, *et al.* Completion time and energy optimization in the UAV-enabled mobile-edge computing system. *IEEE Internet of Things Journal*, Vol. 7, No. 8, pp. 7808–7822, 2020.

[43] L. Zhu and N. Laptev. Deep and confident prediction for time series at Uber. *Proc. of ICDMW*, IEEE, 2017.

[44] T. Zhang, Y. Xu, J. Loo, *et al.* Joint computation and communication design for UAV-assisted mobile edge computing in IoT. *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 8, pp. 5505–5516, 2020.

[45] G. Zhao, H. Xu, Y. Zhao, *et al.* Offloading dependent tasks in mobile edge computing with service caching. *Proc. of INFOCOM*, IEEE, 2020.

[46] L. Zhu, J. Zhang, Z. Xiao, *et al.* Millimeter-wave full-duplex UAV relay: Joint positioning, beamforming, and power control. *IEEE Journal on Selected Areas in Communications*, Vol. 38, No. 9, pp. 2057–2073, 2020.