User Preference Oriented Service Caching and Task Offloading for UAV-assisted MEC Networks

Ruiting Zhou, *Member, IEEE,* Yifeng Huang, Yufeng Wang, Lei Jiao, *Member, IEEE,* Haisheng Tan, *Senior Member, IEEE,* Renli Zhang, and Libing Wu

Abstract—Unmanned aerial vehicles (UAVs) have emerged as a new and flexible paradigm to offer low-latency and diverse mobile edge computing (MEC) services for user equipment (UE). To minimize the service delay, caching is introduced in UAV-assisted MEC networks to bring service contents closer to UEs. However, UAV-assisted MEC is challenged by the heavy communication overhead introduced by service caching and UAV's limited energy capacity. In this paper, we propose an online algorithm, *OOA*, that jointly optimizes caching and offloading decisions for UAV-assisted MEC networks, to minimize the overall service delay. Specifically, to improve the caching effectiveness and reduce the caching overhead, *OOA* employs a greedy algorithm to dynamically make caching decisions based on UEs' preferences on services and UAVs' historical trajectories, with the goal of maximizing the probability of successful offloading. To realize the rational utilization of energy from a long-term perspective, *OOA* decomposes the online problem into a series of single-slot problems by scaling the UAV's energy constraint into the objective, and iteratively optimizes UAV trajectory and task offloading at each time slot. Theoretical analysis proves that *OOA* converges to a suboptimal solution with polynomial time complexity. Extensive simulations based on real world data further show that *OOA* can reduce the service delay by up to 33% while satisfying the UAV's energy constraint, compared to three state-of-the-art algorithms.

Index Terms—Unmanned aerial vehicles, Mobile edge computing, Service caching, Task offloading.

1 Introduction

The rapid development of 5G promotes the application of computing-intensive and data-driven smart services, such as online games and automatic driving [1]. To meet the computing power and low-latency requirements of smart services, mobile edge computing (MEC) has been proposed to transfer the computing tasks of user equipment (UE) to the network edge for processing [2]. Unfortunately, MEC servers need to be deployed on fixed infrastructure, such as base stations (BSs), which incurs high deployment costs. MEC may not be able to work effectively in rural areas that lack sufficient infrastructures or urban areas during peak hours/natural disasters. In recent years, Unmanned Aerial Vehicles (UAVs) equipped with MEC servers have been widely discussed in industry and academia [3]. UAVs can provide low-latency and more flexible computing services due to their high mobility and flexible deployment. For example, on July 21, 2021, Mihe town in China was flooded due to heavy rainfall. In its communication interruption area, 255 UAVs form mobile base stations to provide communication signal that covers 50 square kilometers for five hours [4].

- R. Zhou is with the School of Computer Science Engineering, Southeast University, Nanjing, China, and the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China. E-mail: {ruitingzhou}@seu.edu.cn.
- Y. Huang, Y. Wang, R. Zhang and L. Wu are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China. E-mail: {yifenghuang, yufengwang, zhang_rl, wu}@whu.edu.cn.
- L. Jiao is with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403 USA. E-mail: {jiao}@cs.uoregon.edu.
- H. Tan is with School of Computer Science and Technology, University of Science and Technology of China, and the Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China, Chinese Academy of Sciences. E-mail: {hstan}@ustc.edu.cn

To fully realize UAV-assisted MEC, both service caching and task offloading are required. On the one hand, it has been shown that some popular content is repeatedly requested by UEs [5], [6]. By caching some popular content to UAVs nearby UEs and reusing the stored content, caching is considered to be an effective approach to reduce the service delay. In addition, many popular applications are data-driven and require caching some service content, such as libraries and machine learning models, on the UAV's edge server [7], [8]. On the other hand, offloading computing-intensive and latency-sensitive tasks to UAVs can solve the shortcomings of UEs in terms of computation resources and battery capacities.

However, service caching and task offloading in UAV-assisted MEC face fundamental challenges. **First,** how to dynamically place and update cache contents in UAVs is challenging. Because one UAV has limited storage space, only some services can be cached. Furthermore, UE's location and preference for services may change over time, but the frequent download of service contents from the remote cloud brings heavy communication overhead. **Second,** although the mobility of the UAV increases its flexibility, its flight trajectory significantly affects the number of UEs that the UAV can serve, and ultimately impacts the service delay. Therefore, UAV trajectory needs to be optimized to facilitate the deployment of UAV-assisted MEC. **Third,** due to the limited computation/communication resource and the battery capacity of a UAV, one UAV can only process a certain number of tasks for a period of time [9], [10].

Making offloading decisions for all tasks requires long-term and rational utilization of the resources and the energy.

Existing service caching or offloading in MEC studies the fixed edge or cellular networks [5], [6], [7], [11], [12], [13]. Their approaches cannot solve the service caching or offloading problem in UAV-assisted MEC, since they didn't consider UAV trajectory planing. In the related studies of UAV-assisted MEC, most studies only study service caching [14], [15], [16] or task offloading [17], [18], [19], [20], [21] alone, and do not consider both. A few papers consider both service caching and task offloading. Their solutions are either in offline scenarios [22], [23], or they only update the cache according to the task offloaded by UEs at the current time, ignoring the historical task information. Their caching decisions are not efficient since users' location and preferences may change over time. The caching decision need to be optimized dynamically from a long-term time scale. We will discuss it in detail in Sec. 2.

In this paper, we model UAV's important features (*i.e.,* mobility and limited energy), and study the problem of dynamic service caching and task offloading in energy-constrained UAV-assisted MEC networks, aiming to minimize the service delay. In order to realize the rational utilization of energy from a long-term perspective, we decouple the problem into a series of single-slot problems by splitting the energy constraint into the objective function. At the beginning of each time slot, we decide whether to update the cache according to the hit rate, *i.e.,* the probability of successful offloading. If the gap between the average hit rate (calculated based on the offloading decision) and the expected hit rate (calculated by the caching decision) reaches the threshold, the cache is updated based on UEs' preferences on services and UAVs' historical trajectories to maximize the expected hit rate. In this way, we improve the caching efficiency by observing both offloading results and UEs' preferences. Furthermore, the frequency of updating the cache can be dynamically adjusted by the threshold to reduce the communication overhead. We then propose an iterative algorithm based on first-order Taylor Expansion and dependent rounding technique to make UAV trajectories and task offloading decisions. Notice that in this paper, service caching and task offloading are jointly optimized within one time slot, not simultaneously. Some of the previous works [23], [24] make service caching and task offloading decisions at the same time. However, making caching decisions and offloading decisions simultaneously causes these papers to fail to consider the historical task information. In contrast, this paper dynamically updates service caching decisions based on historical task information, which improves the hit rate and also reduces the cache update frequency. We highlight our contributions as follows.

- We formulate the user preference oriented service caching and task offloading problems in UAV-assisted MEC. The service caching, UAV trajectory and task offloading decisions need to be jointly optimized and made online under energy and resource capacity constraints. To minimize the service delay, the cache is dynamically updated based on the preferences of UEs for services. Both the service caching problem and the task offloading problem are NP-hard.
- We propose an online algorithm, *OOA*, that dynamically updates caching decisions and optimizes UAV trajectory and task offloading. For service caching, *OOA* reformulates the caching problem into a submodular maximization problem and employs a greedy caching algorithm that places services according to the service preferences of UEs. To achieve low latency in the long-term, we splits the energy consumption constraint into the optimization objective by using a weighing factor. The task offloading problem is then decomposed into a series of single-slot problems. *OOA* then develops an iterative algorithm to optimize UAV trajectory and task offloading.
- We conduct rigorous theoretical analysis to prove that *OOA* converges to a suboptimal solution with polynomial time complexity. Moreover, we conduct extensive simulations based on real world data. Simulations results verify that *OOA* achieves near-optimal service delay under strict energy constraint. Particularly, *OOA* can reduce the service delay by up to 33% with 66% energy consumption less than three benchmark algorithms.

In the rest of the paper, we review related work in Sec. 2. The system model of UAV-assisted MEC is introduced in Sec. 3. Our online algorithm, *OOA*, is proposed in Sec. 4. The performance of *OOA* is evaluated in Sec. 5 and Sec. 6 concludes the paper.

2 Related Work

2.1 Service Caching

Xu *et al.* [7] investigate service caching in MEC network and develop a game-theoretical mechanism for resource sharing among service providers. In [11] and [12], the authors focus on the cooperative caching in edge computing via distributed online learning. Caching contents considering users' content preferences is studied in [5], [6], [13]. However, these studies can't be applied to UAV-assisted MEC directly due to the mobility and energy limit of UAVs. Caching in UAVs has also been studied in recent years. Gu *et al.* [14] consider content caching, UAV deployments, and transmitting power allocation in Satellite-UAV-Vehicle-Integrated Networks. Zhang *et al.* [15] use a dynamic UAV trajectory scheduling algorithm to maximize the caching duration. Li *et al.* [16] minimize expected user delay by using mean field game theory to model caching and trajectory problems. Most of the above papers either consider caching a fixed number of services, assume the size of different services is the same, or fail to give rigorous proof for the theoretical performance of the caching algorithm. Besides, these papers only consider caching and can't make online offloading decisions. Different from these papers, we propose an online algorithm that jointly optimizes service caching, UAV trajectory, and task offloading.

2.2 Task Offloading

Wang *et al.* [17] design an iterative cooperation algorithm to dynamically determine multiple UAVs' trajectories, seeking the maximization of the number of served demands. Ning *et al.* [18] aim to maximize the throughput of UAVs by optimizing UAV trajectory and task scheduling. Zhu *et al.* [19] model the offloading of tasks from a cellular network to a UAV cloudlet as a Markov decision process to minimize the average response time. Both Sun *et al.* [20] and Tang *et al.* [21] jointly optimize the UAV trajectory, resource allocation, and task offloading to minimize the energy consumption of the UAV. However, the above papers ignore the problem of service caching. It's crucial to study the service caching on UAVs with restricted storage space, since different tasks may demand different services stored on UAVs.

2.3 Joint Service Caching and Task Offloading

In [22] and [25], the authors estimate content popularity and deploy UAVs to minimize the request delay. Both Ji *et al.* [26] and Zhang *et al.* [27] consider jointly optimizing UAV trajectory, caching placement, and transmitting power to implement content delivery. Nevertheless, these papers can't satisfy the energy limit of UAVs. Wu *et al.* [28] presents a CNN-based model to make online caching and offloading decisions, yet fails to consider the communication and computation resource limits of UAVs. Qu *et al.* [23] optimizes service caching, UAV trajectory, computation resource allocation, and task scheduling simultaneously to minimize energy consumption. However, [23] is designed for offline. Zhou *et al.* [24] develop a two-time-scale online service caching and task offloading algorithm for UAV-assisted networks. But [24] updates caching decisions periodically according to tasks received at one time slot, and can't strictly satisfy the energy budget of UAVs. In this paper, we make dynamic service caching, UAV trajectory, and task offloading decisions under strict energy budget constraints. Considering the time-varying services preferences and locations of UEs, we dynamically update caching decisions according to the preferences of UEs for services. What's more, we consider using the energy of UAVs in a reasonable way to minimize the service delay in the long term.

3 System Model

3.1 System Overview

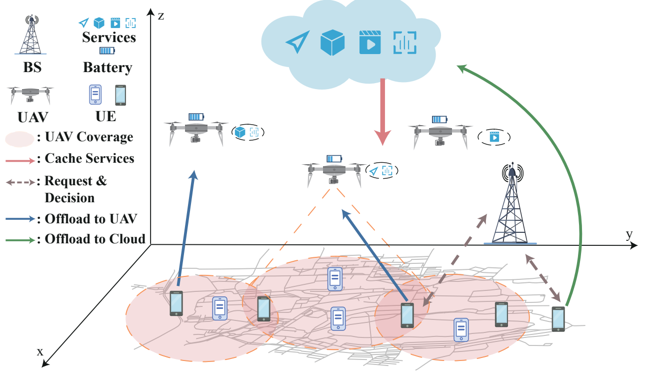


Fig. 1: An illustration of UAV-assisted MEC.

UAV-assisted MEC Scenario. As shown in Fig. 1, we consider a UAV-assisted MEC system with U rotary-wing UAVs, a base station (BS) and a remote cloud. The system provides S types of services for I terrestrial UEs. Let $\mathcal{X}$ denote the integer set $\{1, 2, \ldots, X\}$. For example, $\mathcal{I}$ denotes $\{1, 2, \ldots, I\}$. The system time span is divided into T time slots. UEs may generate tasks requiring services at any time slot. Each task requires one specific type of service. To satisfy the delay requirement, all tasks must be processed at one time slot. However, the BS may be overloaded in some scenarios. Thus, some tasks need to be offloaded to the UAVs or the remote cloud. The remote cloud provides ample computing power for all kinds of services while resulting in high communication delay. In order to reduce the delay, services are cached in UAVs in advance. At the beginning of each time slot, BS decides whether to update the cached services in UAVs or not. If so, UEs firstly upload their personal preferences for services to the BS, and then the BS makes caching decisions according to UEs' interests. Next, the BS collects tasks from UEs and decides whether to offload them to UAVs or to the remote cloud.

Offloading Requirement. Tasks from UEs can be offload to UAVs only if following two conditions are satisfied: i) the UAV has enough communication and computation resources to process the task; ii) the service that the UE's task requires is cached in the UAV. Otherwise tasks will be offloaded to the remote cloud.

3.2 Service Caching

Caching Strategy. In order to reduce the computation delay, services are pre-cached on UAVs. Let W_u be the storage capacity of UAV u, and the storage space required by service s is denoted by c_s. Due to the preferences of UEs are time-varying, the cache decisions should be dynamic updated.

UE Preference. The intuition in designing efficient caching strategy is to cache services according to the preference of UEs. Let $P_{i,s} \in [0, 1]$ be the probability of UE i generating a task requiring service s. Since one UE can generate multiple tasks at the same time, the expectation of the number of tasks generated by UE i is $\sum_i P_{i,s}$. At first, $P_{i,s}$ is uploaded by UEs. As the BS processes more tasks from UEs, the BS will update $P_{i,s}$ based on the historical information of tasks.

Decision Variables and Hit Ratio. The BS decides which services to cache. Let $x_{u,s}$ denote whether service s is cached in UAV u ($x_{u,s} = 1$) or not ($x_{u,s} = 0$). Similar to [5], [29], [30], the hit ratio is used to measure the performance of a caching strategy. The hit ratio of UE i is the probability that tasks generated by UE i can be offloaded to UAVs:

$$H_{i,avg} = \frac{\sum_s P_{i,s}\left(1 - \prod_{u=1}^{U}(1 - \rho_{i,u}x_{u,s})\right)}{\sum_s P_{i,s}},$$

where $\rho_{i,u}$ is the probability that UE i is within the covering radius of UAV u. $\rho_{i,u}$ is calculated according to the UAVs' historical trajectories.

Caching Problem Formulation. We aim to maximize the sum of the hit ratio of UEs. The caching problem can be formulated as:

$$(\mathbf{P1}) \quad \max \textstyle\sum_{i\in\mathcal{I}} H_{i,avg}$$

$$\textstyle\sum_{s\in\mathcal{S}} c_s x_{u,s} \le W_u, \forall u, \tag{1a}$$

$$x_{u,s} \in \{0, 1\}, \forall u, \forall s. \tag{1b}$$

Constraint (1a) represents the storage constraint of UAV u.

Challenges. (**P1**) is a special case of multi-dimension knapsack problems, which is known to be NP-hard [31]. Moreover, according to [32], it's hard to find an efficient polynomial time approximation scheme (EPTAS)[1].

3.3 Task Offloading

Task Information. Let $d^t_{i,s}$ be the size of task requiring service s generated by UE i at slot t, and μ_s be the workload (in terms of CPU cycles) of service s per unit data.

UAV Properties. We consider a 3-D Cartesian coordinate system, in which the horizontal coordinate of UAV u at time slot t is denoted as $G^t_u = (X^t_u, Y^t_u)$ (we assume that UAV u flies at a fixed altitude Z_u). The horizontal distance between UE i and UAV u at time slot t is calculated as: $D^t_{i,u} = \sqrt{(X^t_i - X^t_u)^2 + (Y^t_i - Y^t_u)^2}$, where $G^t_i = (X^t_i, Y^t_i)$ is the horizontal coordinate of UE i at time slot t. Following [33], the uplink data rate of UE i with UAV u at time slot t is calculated as:

$$r^t_{u,i} = B\log_2\left(1 + \frac{\delta p_u}{\left(D^t_{i,u}\right)^2 + Z^2_u}\right),$$

where B is the bandwidth, δ is SNR, and p_u is the transmission power of UAV u. Tasks from UE i can be offloaded to UAV u only when UE i is within the UAV's maximum horizontal covering radius R^{max}_u. Meanwhile, suppose that the UAV flies at a constant speed, therefore the maximum flight distance of UAV u between two time slots is limited, denoted as D^{max}_u. Due to the communication and computation resource limits of UAVs, UAV u can only process at most N^{max}_u tasks at one time slot.

Decision Variables. Suppose that each task can be offloaded to at most one UAV or the remote cloud. Following decisions need to be made jointly by the BS at each time slot: i) $a_t \in \{0,1\}$, denotes whether the services cached in UAVs will be updated at time slot t; ii) $y^t_{i,u,s}(y^t_{i,0,s}) \in \{0,1\}$, represents whether the task from UE i requiring service s is offloaded to UAV u (the remote cloud) at time slot t; iii) $G^t_u = (X^t_u, Y^t_u)$, the horizontal coordinate of UAV u at time slot t.

Service Delay. The delay experienced by UEs when offloading tasks consists of two parts: i) *Communication Delay:* The communication delay includes the delay to upload the task and the delay to return the computation result. Since the size of the computation result is generally much smaller than the task's input size, the delay to return the computation result is ignored here. Thus the communication delay can be calculated as: $L^t_{i,comm} = \sum_s(\frac{d^t_{i,s}y^t_{i,0,s}}{r_0} + \sum_u \frac{d^t_{i,s}y^t_{i,u,s}}{r^t_{u,i}})$, where r_0 is the average uplink data rate of UE with the remote cloud. ii) *Computation Delay:* The remote cloud is assumed to have sufficient computing power, hence the computation delay of the remote cloud can be neglected. Let f_u denote the computing capacity (in terms of CPU cycles) of UAV u for each task. Therefore the computation delay of UE i at time slot t can be calculated as: $L^t_{i,comp} = \sum_s \sum_u \frac{d^t_{i,s}\mu_s y^t_{i,u,s}}{f_u}$. The overall delay of UE i at time slot t is calculated as:

$$L^t_i = L^t_{i,comm} + L^t_{i,comp}.$$

Energy Consumption of UAVs. The battery capacity for UAV u is limited, denoted as E^{max}_u. We consider four types of energy consumption for UAV u at time slot t. i) *Communication Energy Consumption:* The communication energy consumption is proportional to the communication delay, thus can be calculated as: $E^t_{u,comm} = \sum_i \sum_s p_u \frac{d^t_{i,s}y^t_{i,u,s}}{r^t_{u,i}}$. ii) *Computation Energy Consumption:* The unit energy consumption per time of UAV u when executing tasks is denoted as γ_u. Thus the computation energy consumption of UAV u at time slot t can be calculated as: $E^t_{u,comp} = \gamma_u \sum_i \sum_s \frac{d^t_{i,s}\mu_s y^t_{i,u,s}}{f_u}$. iii) *Flight Energy Consumption:* Let ω_u denote the unit flying energy consumption of UAV u. The flight energy consumption can be calculated as: $E^t_{u,fly} = \omega_u \left\|G^t_u - G^{t-1}_u\right\|$. iv) *Caching Energy Consumption:* The caching energy consumption is caused by storing services. Let η denote the unit energy consumption per data for storing. The caching services set of UAV u at time slot t is represented by S^t_u. The caching energy consumption can be calculated as $E^t_{u,cach} = \sum_{s\in S^t_u/S^{t-1}_u} \eta c_s$. The overall energy consumption of UAV u at time slot t is calculated as:

$$E^t_u = E^t_{u,comm} + E^t_{u,comp} + E^t_{u,fly} + a^t E^t_{u,cach}.$$

Important notations are listed in Table I for easy reference.

TABLE 1. Notations

$U/\mathcal{U}$	number/set of UAVs
$I/\mathcal{I}$	number/set of UEs
$S/\mathcal{S}$	number of types/set of services
$T/\mathcal{T}$	number/set of time slots
(X^t_u, Y^t_u)	horizontal coordinate of UAV u at time slot t
R^{max}_u	maximum coverage radius of UAV u
$D^t_{i,u}$	distance between UE i and UAV u
$r^t_{u,i}$	uplink date rate between UE i and UAV u at time t
N^{max}_u	number of tasks UAV u can process at one time slot
W_u	storage capacity of UAV u
c_s	storage space required by service s
$P_{i,s}$	probability of UE i generating a task requiring service s
$d^t_{i,s}$	size of task generated by UE i at time t for service s
$x_{u,s}$	whether the service s is cached at UAV u
$y^t_{i,u,s}$	whether UE i's task requiring service s is offloaded to UAV u at time slot t

Offloading Problem Formulation. The task offloading problem is formulated as:

$$(\textbf{P2}) \quad \min \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{I}} L^t_i$$

$$y^t_{i,u,s}\left(D^t_{i,u}\right)^2 \le \left(R^{\max}_u\right)^2, \forall i, \forall u, \forall s, \forall t \tag{2a}$$

$$y^t_{i,u,s} \le x_{u,s}, \forall i, \forall u, \forall s, \forall t, \tag{2b}$$

$$\sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}} y^t_{i,u,s} \le N^{max}_u, \forall u, \forall t, \tag{2c}$$

$$\sum_{u=0}^{U} y^t_{i,u,s} = 1, \forall i, \forall s, \forall t, \tag{2d}$$

$$\sum_{t\in\mathcal{T}} E^t_u \le E^{max}_u, \forall u, \tag{2e}$$

$$\left\|G^{t+1}_u - G^t_u\right\| \le D^{max}_u, \forall u, \forall t, \tag{2f}$$

$$a_t \in \{0,1\}, y^t_{i,0,s} \in \{0,1\}, y^t_{i,u,s} \in \{0,1\}, \forall u, \forall s, \forall t. \tag{2g}$$

Constraint (2a) and (2b) ensure that the task can be offload only if the UE is within the coverage radius of UAV and the UAV has cached the corresponding service. Constraint (2c) limits the maximum number of tasks offloading to UAV u at one time slot. Constraint (2d) ensures that tasks generated

1. An EPTAS is a polynomial time $(1-\epsilon)$-approximation scheme whose running time is $f(\frac{1}{\epsilon})|\mathcal{I}|^{O(1)}$ for some function f and any input $\mathcal{I}$.

at time slot t must be offloaded at the current time slot. Constraint (2e) is UAV u's energy constraint, where E_u^{max} is the battery capacity of UAV u. Constraint (2f) captures the maximum fly distance of UAV between two time slots ($\|\cdot\|$ is the L2-norm).

Challenges. i) The offloading requests arrive online, and the BS has to make offloading decisions on the fly. ii) (**P2**) is a mixed-integer nonlinear programming (MINLP). Even in the offline setting, (**P2**) is NP-hard (With fixed UAV trajectories, the problem is a multi-dimension knapsack problem, which is NP-hard [31]). iii) Constraint (2e) in (**P2**) involves all time slots while other constraints refer to one time slot, which further poses challenges in the algorithm design. iv) The solution of (**P1**) will affect (**P2**). The coupling between (**P1**) and (**P2**) makes the difficulty of the problem further escalated.

4 Online Algorithm Design

4.1 Algorithm Idea and Overall Algorithm

In this section, we design an online algorithm, *OOA*, that dynamically updates services caching and determines task offloading at each time slot to minimize the overall service delay. Fig. 2 shows the main idea.

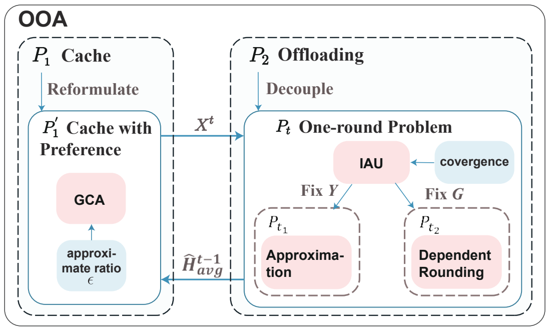


Fig. 2: Main Idea of *OOA*.

i. At the beginning of each time slot, *OOA* decides whether to update services cached in UAVs based on the gap between the average hit ratio of previous offloaded tasks and the expected hit ratio calculated by the caching decision. If so, *OOA* solves **P1** by a greedy algorithm *GCA* according to UEs' preferences for services and UAVs' historical trajectories. Then, by scaling the energy consumption into the objective function using an energy weighting factor, **P2** is decomposed into one-slot problems $\mathbf{P_t}$. Next, *OOA* solves $\mathbf{P_t}$ by an iterative algorithm *IAU* to obtain task offloading decisions for time slot t.
ii. For caching problem **P1**, we first reformulate the origin problem by replacing the objective function with an equivalent submodular function H. Then, the algorithm *GCA* greedily makes caching decisions to maximize the sum of the hit ratio. The theoretical performance of the algorithm *GCA* is guaranteed, owning to the submodularity of H.
iii. For one-slot task offloading problem $\mathbf{P_t}$, we propose an iterative algorithm *IAU* that alternately optimizes UAV trajectory and task offloading. First, *IAU* fixes task offloading decsions Y to obtain subproblem $\mathbf{P_{t_1}}$, which is non-convex. To solve $\mathbf{P_{t_1}}$, $\mathbf{P_{t_1}}$ is converted to a simplified problem $\mathbf{P'_{t_1}}$ by replacing the non-convex part in the objective function with an upper bound. $\mathbf{P'_{t_1}}$ is a convex problem and can be solved by standard convex solver CVX [34]. Second, with the fixed UAV trajectory, *IAU* obtains subproblem $\mathbf{P_{t_2}}$, which is a 0-1 integer linear programming problem. We first relax the variables in $\mathbf{P_{t_2}}$ to fractional and get the fractional solution by interior point method. Then we use a dependent rounding algorithm *DR* to round the fractional solution into the integer solution.

Overall Algorithm Details. We summarize our algorithm *OOA* in Alg. 1. Caching decisions and the expected hit ratio $H_e = \frac{\sum_i H_{i,avg}}{I}$ are initialized according to Alg. 2 before offloading tasks (line 2). At the beginning of each time slot $t > 1$, if the average hit ratio $\hat{H}_{avg}^{t-1}$ is significantly smaller than H_e, the cached services in UAVs, as well as the expected hit ratio, will be updated (lines 5-8)[2]. Then UAV trajectory and task offloading decisions are made according to Alg. 4 (line 12). At the end of the time slot, the energy weighting factor and the average hit ratio $\hat{H}^t$ are updated (lines 13-14).

Algorithm 1 Overall Online Algorithm (OOA).

Input: Hit ratio tolerance κ, energy weighting factor α
Output: Service caching X^t, UAV trajectory $\mathbf{G}^t$ and task offloading $\mathbf{Y}^t, \forall t \in \mathcal{T}$

```
1: Initialize $\lambda_u^t = 0, a_t = 0, t_a = 1, \forall t, \forall u$;
2: Initialize caching decisions $X^0$ and the expected hit ratio $H_e$ according to Alg. 2;
3: for $t = 1 : T$ do
4:     if $t \neq 1$ and $H_e - \hat{H}_{avg}^{t-1} > \kappa$ then
5:         Set $a_t = 1, t_a = t$;
6:         Update $P_{i,s}$ according to Eq. (3)
7:         Update $X^t$ according to Alg. 2;
8:         Update $H_e$ based on $X^t$;
9:     else
10:        Update $X^t = X^{t-1}$;
11:    end if
12:    Obtain $\mathbf{G}^t, \mathbf{Y}^t$ according to Alg. 4;
13:    Update $\lambda_u^t$ according to Eq. (6);
14:    Update $\hat{H}_{avg}^t$ according to Eq. (4);
15: end for
```

Let $q_{i,s,t}$ be the number of tasks requiring service s generated by UE i until slot t, and $P_{i,s}^*$ be the task preference uploaded by UE i. Considering both the historical information of tasks and task preference uploaded by UEs, $P_{i,s}$ after time slot t can be calculated as:

$$P_{i,s} = \frac{t}{T} \cdot \frac{q_{i,s,t}}{T} + (1 - \frac{t}{T}) P_{i,s}^*. \tag{3}$$

Let $\Upsilon^t = \{(i,s) | d_{i,s}^t > 0, \forall i \in \mathcal{I}, \forall s \in \mathcal{S}\}$ be the task set at time slot t, $\Upsilon_0^t = \{(i,s) | d_{i,s}^t > 0 \wedge y_{i,0,s}^t = 1, \forall i \in \mathcal{I}, \forall s \in \mathcal{S}\}$ be the task set that is offloaded to the remote cloud at time slot t, and t_a be the time slot when the caching was last updated. Then, $\hat{H}_{avg}^t$ are calculated as:

$$\hat{H}_{avg}^t = 1 - \frac{\sum_{t'=t_a}^{t-1} |\Upsilon_0^{t'}|}{\sum_{t'=t_a}^{t-1} |\Upsilon^{t'}|}. \tag{4}$$

4.2 Caching Algorithm

Reformulation. Let $Q = U \times S$ denote the caching ground set, where $(u, s) \in Q$ indicates that UAV u caches service s. We define a set function H on subsets Ψ of Q:

2. The value of the hit ratio tolerance κ can be adjusted to fit different scenarios.

$$H(\Psi)=\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}}P_{i,s}\big(1-\prod_{u\in\Psi(s)}(1-\rho_{i,u})\big)/(\sum_{i}P_{i,s}),$$

where $\Psi(s)=\{u|(u,s)\in\Psi\}$. Denote $\Psi[u]=\{s|(u,s)\in\Psi\}$, then problem **P1** can be reformulated as:

$$(\mathbf{P1}')\quad \max_{\Psi\subseteq Q} H(\Psi)$$

$$\textstyle\sum_{s\in\Psi[u]}c_s\le W_u,\forall u\in\mathcal{U}. \tag{5a}$$

Lemma 1. *Function H is a monotone non-increasing submodular function.*

Proof. We first proof that H is monotone non-increasing. Define $H(X|\Psi)=H(\Psi\cup X)-H(\Psi)$. For $\Psi\subseteq Q$ and $(u_0,s_0)\in Q\backslash\Psi$, we have $H\big((u_0,s_0)|\Psi\big)=\sum_{i\in\mathcal{I}}P_{i,s_0}\Big(\rho_{i,u_0}x_{u_0,s_0}\prod_{u\in\Psi(s_0)}(1-\rho_{i,u}x_{u,s})\Big)\ge 0$. Next we proof the submodularity of H. For $\Psi_1\subseteq\Psi_2,\Psi_2\subseteq Q$, and $(u_0,s_0)\in Q\backslash\Psi_2$, we have $H\big((u_0,s_0)|\Psi_1\big)-H\big((u_0,s_0)|\Psi_2\big)=\sum_{i\in\mathcal{I}}P_{i,s_0}\Big(\rho_{i,u_0}x_{u_0,s_0}\Big(1-\prod_{u\in\Psi_2(s_0)\backslash\Psi_1(s_0)}(1-\rho_{i,u}x_{u,s})\Big)\times\prod_{u\in\Psi_1(s_0)}(1-\rho_{i,u}x_{u,s})\Big)\ge 0$, which concludes the lemma. □

Based on the submodularity of H, we present a greedy algorithm *GCA* to solve problem **P1**$'$ with theoretical performance guarantee. The key idea is to progressively select UAV-service pair that brings the highest marginal increase in the sum of the hit ratio while satisfying the storage constraints.

Algorithm 2 Greedy Caching Algorithm Design (GCA)

```
Input: $c_s, W_u, \rho_{i,u}, P_{i,s}, \forall i\in\mathcal{I}, \forall u\in\mathcal{U}, \forall s\in\mathcal{S}$
Output: $x_{u,s}, \forall u\in\mathcal{U}, \forall s\in\mathcal{S}$
1: Initialize $\Psi=\emptyset, V=\{(u,s)|u\in\mathcal{U}, s\in\mathcal{S}\}, x_{u,s}=0, W'_u=W_u, \forall u\in\mathcal{U}, \forall s\in\mathcal{S}$;
2: while $|V|>0$ do
3:     $(\hat{u},\hat{s})=\arg\max_{(u,s)\in V}\frac{H(\Psi\cup(u,s))-H(\Psi)}{c_s}$;
4:     $\Psi=\Psi\cup(\hat{u},\hat{s})$;
5:     $V=V\backslash(\hat{u},\hat{s})$;
6:     $x_{\hat{u},\hat{s}}=1, W'_{\hat{u}}=W'_{\hat{u}}-c_{\hat{s}}$;
7:     for $(\hat{u},s)\in V$ do
8:         if $c_s>W'_{\hat{u}}$ then
9:             $V=V\backslash(\hat{u},s)$;
10:        end if
11:    end for
12: end while
```

Caching Algorithm Details. The algorithm *GCA* is described in Alg. 2. First, *GCA* initializes the greedy solution set Ψ, available UAV-service pair set V and remaining storage space W'_u for each UAV (line 1). Then, in each loop, *GCA* successively selects (u,s) pair in V with the highest increment for the objective function H per unit storage cost (line 3). The pair chosen in line 3 is added into Ψ and removed from V (lines 4-5). The remaining storage space of the UAV is updated in line 6. After adding a pair into Ψ, pairs that do not satisfy the storage constraint are removed from V (lines 7-11).

Now we analyze the theoretical performance of Alg. 2. Previous works about service caching also apply greedy algorithm to solve submodular function maximization problems [5]. However, the proofs of their algorithms are valid only when the size of different services is the same. We consider different size of the cached services and give proof of the approximation ratio in this case. The approximation ratio of Alg. 2 is given in Theorem. 1.

Theorem 1. *Let $k=\frac{\min_u W_u}{\max_s c_s},\epsilon=\frac{k}{k-1}$, Alg. 2 is $(1-e^{-\epsilon})$-approximate algorithm to problem **P1**.*

Proof. Let X_i denote the *i-th* element picked by Line 3 in Alg. 2. Let $\Psi_i=(X_1,X_2,....X_i)$ denote the greedy solution set after i-th picking. The greedy solution is defined as $\Psi_l=(X_1,X_2,....X_l)$ (the greedy algorithm picked l elements before it stops). $c()$ is the storage cost function. Let OPT denote the optimal solution of problem **P1**. Let $B=\sum_u W_u$, base on the definition of greedy rule and k, we have $\sum_{i=1}^{l}c(X_i)>=(1-\frac{1}{k})B, L\le B$. Using Lemma 3 in [35], we have

$$\begin{aligned}H(G_l)&\ge\left(1-\prod_{k=1}^{l}\left(1-\frac{c(X_k)}{L}\right)\right)H(OPT)\\&\ge\left(1-\prod_{k=1}^{l}\left(1-\frac{c(X_k)}{B}\right)\right)H(OPT)\\&\ge\left(1-\prod_{k=1}^{l}\left(1-\frac{\sum_{i=1}^{l}c(X_i)}{Bl}\right)\right)H(OPT)\\&\ge\left(1-\prod_{k=1}^{l}\left(1-\frac{(1-\frac{1}{k})B}{Bl}\right)\right)H(OPT)\\&=\left(1-\left(1-\frac{\epsilon}{l}\right)^{l}\right)H(OPT)\ge(1-e^{-\epsilon})H(OPT)\end{aligned}$$

The first inequality is the Lemma 3 in [35]. The second inequality is trivial since $L\le B$. The third inequality holds because the inequality of arithmetic and geometric means. The fourth inequality is due to $\sum_{i=1}^{l}c(X_i)\ge(1-\frac{1}{k})B$. The last inequality is true since $g(x)=1-(1-\frac{\epsilon}{x})^x,\epsilon>0$ is monotone decreasing in $(0,+\infty)$ and $\lim_{x\to+\infty}g(x)=1-e^{-\epsilon}$. □

4.3 Offloading Algorithm

One-slot Offloading Problem. To eliminate the coupling of energy consumption in constraint (2e), for each UAV u, we design an energy weighting factor λ_u^t to decompose problem **P2** into one-slot problems. How the energy budget of UAVs is spent will significantly affect the total service delay in the entire period. Running out of the energy budget of UAVs too soon will limit the future decision space for task offloading. The BS has to select UAVs with higher delay in the later stage, which further increases the service delay in the whole time span. Based on the analysis above, the energy weighting factor should increase as the remaining energy of the UAV decreases. Besides, the factor should have lower and upper bounds, which represent two extreme cases, *i.e.*, no energy usage and energy exhaustion. According to these characteristics, we design λ_u^t which satisfies forementioned requirements as follows:

$$\lambda_u^t=\frac{\bar{L}^t\sum_{t'=1}^{t-1}E_u^{t'}}{(E_u^{max})^2},\forall t\in\mathcal{T}, \tag{6}$$

where $\bar{L}^t=\frac{\sum_{t'=1}^{t-1}\sum_{i\in\mathcal{I}}L_i^{t'}}{\sum_{t'=1}^{t-1}|\Upsilon^{t'}|}$ is the average service delay of tasks before time slot t. The initial value of λ_u^t is zero. λ_u^t increases when the remaining energy of the UAV goes down, and reaches the maximum value when the UAV has no energy. Then the one-slot task offloading problem is formulated as:

$$(\mathbf{P_t}) \quad \min \textstyle\sum_{i\in\mathcal{I}} L_i^t + \sum_{u\in\mathcal{U}} \lambda_u^t E_u^t$$

$$y_{i,u,s}^t \left(D_{i,u}^t\right)^2 \le (R_u^{max})^2, \forall i, \forall u, \forall s \tag{7a}$$

$$y_{i,u,s}^t \le x_{u,s}, \forall i, \forall u, \forall s \tag{7b}$$

$$\textstyle\sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}} y_{i,u,s}^t \le N_u^{max}, \forall u, \tag{7c}$$

$$\textstyle\sum_{u=0}^{U} y_{i,u,s}^t = 1, \forall i, \forall s, \tag{7d}$$

$$E_u^t \le E_{u,r}^t, \forall u, \tag{7e}$$

$$\left\|G_u^{t+1} - G_u^t\right\| \le D_u^{max}, \forall u, \tag{7f}$$

$$y_{i,0,s}^t \in \{0,1\}, y_{i,u,s}^t \in \{0,1\}, \forall u, \forall s. \tag{7g}$$

where $E_{u,r}^t = E_u^{max} - \sum_{t'=1}^{t-1} E_u^{t'}$ is the remaining energy of UAV u at the beginning of time slot t.

We further decompose problem $\mathbf{P_t}$ into two subproblems, *i.e.*, UAV trajectory and task offloading.

4.3.1 UAV Trajectory

With the fixed task offloading decisions, the UAV trajectory subproblem is formulated as:

$$(\mathbf{P_{t_1}}) \quad \min \textstyle\sum_{i\in\mathcal{I}} L_{i,comm}^t + \sum_{u\in\mathcal{U}} \lambda_u^t (E_{u,comm}^t + E_{u,fly}^t)$$
$$\text{s.t.} \quad \text{(7a), (7e), (7f).}$$

Problem $\mathbf{P_{t_1}}$ is non-convex in G_u^t due to the logarithmic part. We try to find an approximation for the non-convex part in problem $\mathbf{P_{t_1}}$ so that the complexity is decreased. It's easy to verify that function $f(x) = \frac{1}{\ln(1+\frac{1}{x})}, \forall x > 0$ is a concave function, thus we have:

$$\tfrac{1}{r_{u,i}^t} \le \tfrac{\ln 2}{B}\big(f'(\zeta)\tfrac{\left\|G_u^t - G_i^t\right\|^2 + Z_u^2}{\delta p_u} - \zeta) + f(\zeta)\big).$$

where $\zeta = \frac{Z_u^2}{\delta p_u}$. Then the simplified UAV trajectory subproblem is given as:

$$(\mathbf{P'_{t1}}) \quad \min \textstyle\sum_i \sum_u \sum_s f'(\zeta) \ln 2 \left(\frac{(1+\lambda_u^t p_u) d_{i,s}^t y_{i,u,s}^t \left\|G_u^t - G_i^t\right\|^2}{B\delta p_u}\right)$$
$$+ \textstyle\sum_u \lambda_u^t \omega_u \left\|G_u^t - G_u^{t-1}\right\|$$
$$\text{s.t.} \quad \text{(7a), (7f),}$$

$$\textstyle\sum_i \sum_s p_u d_{i,s}^t y_{i,u,s}^t \frac{\ln 2}{B}\big(f'(\zeta)(\frac{\left\|G_u^t - G_i^t\right\|^2 + Z_u^2}{\delta p_u} - \zeta) + f(\zeta)\big)$$
$$+ \omega_u \left\|G_u^t - G_u^{t-1}\right\| \le E_{u,r}^t - (E_{u,comp}^t + a^t E_{u,cach}^t), \forall u.$$

Problem $\mathbf{P'_{t1}}$ is convex on G_u^t and can be solved by CVX. In this way, the UAV trajectory subproblem is solved.

4.3.2 Suboptimal Task Offloading

With fixed UAV trajectory, we merge constraint (7a) and constraint (7b) into one constraint, then the task offloading problem is rewritten as:

$$(\mathbf{P_{t_2}}) \quad \min \textstyle\sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}} \sum_{u\in\mathcal{U}} \left(\frac{1+\lambda_u^t p_u}{r_{u,i}^t} + \frac{(1+\lambda_u^t \gamma_u)\mu_s}{f_u}\right.$$
$$\left. - \tfrac{1}{r_0}\right) d_{i,s}^t y_{i,u,s}^t$$

$$y_{i,u,s}^t \le \mathbb{1}_{\ge 1}\left(\min(\tfrac{(R_u^{max})^2}{(D_{i,u}^t)^2}, x_{u,s})\right), \forall i, \forall u, \forall s \tag{11a}$$

$$\textstyle\sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}} y_{i,u,s}^t \le N_u^{max}, \forall u, \tag{11b}$$

$$\textstyle\sum_{u\in\mathcal{U}} y_{i,u,s}^t \le 1, \forall i, \forall s, \tag{11c}$$

$$\textstyle\sum_i \sum_s (\frac{p_u}{r_{u,i}^t} + \frac{\gamma_u \mu_s}{f_u}) d_{i,s}^t y_{i,u,s}^t \le E_{u,r}^t - E_{u,fly}^t - a_t E_{u,cach}^t, \forall u, \tag{11d}$$

$$y_{i,u,s}^t \in \{0,1\}, \forall i, \forall u, \forall s, \tag{11e}$$

where $\mathbb{1}(x) = 1$ if $x \ge 1$, otherwise $\mathbb{1}(x) = 0$. By relaxing all integral decision variables to fractional, problem $\mathbf{P_{t_2}}$ then becomes a continuous linear program $\mathbf{P'_{t_2}}$.

We first use interior point method [36] to obtain a fractional solution of problem $\mathbf{P'_{t2}}$. Then we introduce a dependent rounding algorithm (*DR*) in Alg. 3 that converts the fractional solution into the integer solution with theoretical performance guarantee. For preparation of the rounding procedure, *DR* first constructs a bipartite graph (A, B, E) based on fractional solution Y^*. The steps to construct bipartite graph are as follows: i) Let $A = \{a_{is} | \forall i \in \mathcal{I}, \forall s \in \mathcal{S}\}$. The node a_{is} in A denotes the task requiring service s from UE i. ii) Let $B = \{b_u | \forall u \in \mathcal{U}\}$. The node b_u in B denotes the UAV u. iii) For $a_{is} \in A$ and $b_u \in B$, put the edge (a_{is}, b_u) into E with weight $e_{is,u} = y_{i,s,u}^{t*}$.

Algorithm 3 Dependent Rounding Algorithm (DR), $\forall t$

```
Input: Fractional solution $\mathbf{Y}^*$
Output: Integer solution $\bar{\mathbf{Y}}$
1:  Construct bipartite graph $(A, B, E)$ based on $\mathbf{Y}^*$;
2:  while $E \neq \emptyset$ do
3:      Remove edges in $E$ such that $e_{is,u} \in \{0,1\}$;
4:      while there exists a cycle or longest path $\Gamma$ do
5:          Divide $\Gamma$ into two matchings $M_1$ and $M_2$;
6:          $\eta_1 \overset{def}{=} \min\{\eta : (\exists(a_{is}, b_u) \in M_1 : e_{is,u} + \eta = 1) \bigvee (\exists(a_{is}, b_u) \in M_2 : e_{is,u} - \eta = 0)\}$.
7:          $\eta_2 \overset{def}{=} \min\{\eta : (\exists(a_{is}, b_u) \in M_1 : e_{is,u} - \eta = 0) \bigvee (\exists(a_{is}, b_u) \in M_2 : e_{is,u} + \eta = 1)\}$.
8:          With the probability $\frac{\eta_2}{\eta_1+\eta_2}$,
            Set $e_{is,u} = e_{is,u} + \eta_1, \forall(a_{is}, b_u) \in M_1$ and $e_{is,u} = e_{is,u} - \eta_1, \forall(a_{is}, b_u) \in M_2$;
9:          With the probability $\frac{\eta_1}{\eta_1+\eta_2}$,
            Set $e_{is,u} = e_{is,u} - \eta_2, \forall(a_{is}, b_u) \in M_1$ and $e_{is,u} = e_{is,u} + \eta_2, \forall(a_{is}, b_u) \in M_2$;
10:     end while
11: end while
12: Set $\bar{y}_{i,u,s}^t = e_{is,u}, \forall i, \forall u, \forall s$;
13: Set $\bar{y}_{i,0,s}^t = 1 - \sum_{u\in\mathcal{U}} \bar{y}_{i,u,s}^t, \forall i, \forall s$;
14: Return $\bar{\mathbf{Y}}$
```

Rounding Algorithm Details. In Alg. 3, *DR* first eliminates edges with integral weights in line 3. Lines 4-10 provide the rounding process. In each iteration, *DR* finds a cycle or longest path and splits it into two matchings (line 5). Then, with carefully designed probability, weights of edges in one matching will increase while weights of edges in the other matching are decreased (lines 6-9). Finally, the integer solution is updated in lines 12-13.

Now we study the theoretical performance of Alg. 3 and whether the integer solution returned by Alg. 3 satisfies constraints of problem $\mathbf{P_{t_2}}$.

Lemma 2. *Let P_{t_2} be the objective function of problem $\mathbf{P_{t_2}}$. Given the fractional solution Y^* of $\mathbf{P_{t_2}}$ and the corresponding integer solution $\bar{Y}$ returned by Alg. 3, we have:*

$$\mathbb{E}(P_{t_2}(\bar{Y})) = \mathbb{E}(P_{t_2}(Y^*)).$$

Proof. We first proof that $\mathbb{E}(\bar{y}_{i,u,s}^t) = y_{i,u,s}^{t*}$. Suppose Alg. 3 stops after J iterations. Let $e_{is,u}^j$ denote the weight of edge (a_{is}, b_u) after j iterations. As a result, we have $e_{is,u}^0 = y_{i,u,s}^{t*}$ and $e_{is,u}^J = \bar{y}_{i,u,s}^t$. Now Consider iteration $j+1$.

Case 1: The edge is not part of the cycle or the longest path, then its weight remains intact.

Case 2: The weight of the edge has been modified after iteration $j+1$, according to lines 8-9 in Alg. 3 we have

$$\mathbb{E}(e_{is,u}^{j+1}) = \frac{\eta_2}{\eta_1+\eta_2}(\mathbb{E}(e_{is,u}^{j})+\eta_1) + \frac{\eta_1}{\eta_1+\eta_2}(\mathbb{E}(e_{is,u}^{j})-\eta_2)$$
$$= \frac{\eta_1}{\eta_1+\eta_2}\mathbb{E}(e_{is,u}^{j}) + \frac{\eta_2}{\eta_1+\eta_2}\mathbb{E}(e_{is,u}^{j}) = E(e_{is,u}^{j}).$$

In both cases $\mathbb{E}(e_{is,u}^{j+1}) = \mathbb{E}(e_{is,u}^{j})$ holds. Thus we have $\mathbb{E}(\bar{y}_{i,u,s}^t) = \mathbb{E}(e_{is,u}^J) = \cdots = \mathbb{E}(e_{is,u}^0) = y_{i,u,s}^{t*}$. Notice that problem $\mathbf{P_{t_2}}$ can be reformulated as $\min_Y \sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}} m_{i,u,s}^t y_{i,u,s}^t$, where $m_{i,u,s}^t = (\frac{1+\lambda_u^t p_u}{r_{u,i}^t} + \frac{(1+\lambda_u^t\gamma_u)\mu_s}{f_u} - \frac{1}{r_0})d_{i,s}^t, \forall i,u,s,t$. Using $\mathbb{E}(\bar{y}_{i,u,s}^t) = y_{i,u,s}^{t*}$, we obtain $\mathbb{E}(P_{t_2}(\bar{Y})) = \mathbb{E}(P_{t_2}(Y^*))$. □

Lemma 3. *Under the solution $\bar{Y}$ returned by Alg. 3, constraint (11a), (11b) and (11c) must be satisfied, while constraint (11d) are in expectation satisfied.*

Proof. It's easy to verify that constraint (11a) must be satisfied since $\mathbb{1}(x) \in \{0,1\}$. As for the constraint (11b) and (11c). Consider a vertex v in the bipartite graph constructed in Alg. 3. At any iteration, it's trivial that (11b) and (11c) hold If v has at most one edge incident on it. Now consider v has at least two edges incident on it. For cycle or longest path Γ and two matchings M_1, M_2, v must have exactly two edges in Γ, and one of them is in M_1 while the other is in M_2. Thus the modification of weights in Alg. 3 won't affect $\sum_u e_{is,u}$ and $\sum_{i,s} e_{is,u}$, *i.e.*, $\sum_u \bar{y}_{i,u,s}^t$ and $\sum_{i,s}\bar{y}_{i,u,s}^t$, hence (11b) and (11c) hold. Finally, we proof that constraint (11e) is satisfied. Constraint (11e) can be rewritten as $\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}} n_{i,u,s}^t y_{i,u,s}^t \le E_{u,r}^t - E_{u,fly}^t - a_t E_{u,cach}^t$, where $n_{i,u,s}^t = \frac{p_u d_{i,s}^t}{r_{u,i}^t} + \frac{\gamma_u d_{i,s}^t}{f_u}$. The following proof is similar to the proof of Theorem. 2 since $\mathbb{E}(\bar{y}_{i,u,s}^t) = y_{i,u,s}^{t*}$. □

Lemma 4. *Alg. 3 converges in polynomial time.*

Proof. Since at least one edge is removed from E per iteration, the **while** loop in lines 2-11 of Alg. 3 terminates in $O(|E|) = O(IUS)$ iterations. For each iteration, a cycle or longest path can be found in $O(|A\cup B|) = O(IS+U)$ steps via Depth First Search(DFS). Lines 8-9, 12-13 in Alg. 3 also take $O(IUS)$ steps. Therefore the running time of Alg. 3 is $O((IS+U)IUS)$. □

4.3.3 Iterative Algorithm Design

Algorithm 4 Iterative Algorithm For UAV Trajectory and Task Offloading (IAU).

```
Input: Tolerance $\phi$, maximum iterations $I^{max} = 100$
Output: Solution $\{\mathbf{G}, \mathbf{Y}\}$
 1: Initialize a feasible solution $\{\mathbf{G}^0, \mathbf{Y}^0\}$, and iteration index $i = 1$;
 2: while $i \le I^{max}$ do
 3:     With fixed $\mathbf{Y}^{i-1}$, obtain $\mathbf{G}^i$ by solving $\mathbf{P'_{t1}}$;
 4:     Fix $\mathbf{G}^i$, obtain fractional solution $\mathbf{Y}^{i*}$ by solving $\mathbf{P'_{t2}}$;
 5:     obtain integer solution $\mathbf{Y}^i$ according to Alg. 3;
 6:     if $|P_t(\mathbf{G}^i, \mathbf{Y}^i) - P_t(\mathbf{G}^{i-1}, \mathbf{Y}^{i-1})| \le \phi$ then
 7:         Break;
 8:     else
 9:         Set $i \leftarrow i + 1$.
10:     end if
11: end while
12: Return $\{\mathbf{G}^i, \mathbf{Y}^i\}$
```

To obtain a suboptimal solution to problem $\mathbf{P_t}$, we develop an alternating optimization-based algorithm (*IAU*) in Alg. 4 that iteratively solves two subproblems. In Alg. 4, *IAU* iteratively optimizes UAV trajectory (line 3) and task offloading (lines 4-5). The complexity and convergence analysis of Alg. 4 is given as follows:

Lemma 5. *Alg. 4 converges with polynomial complexity.*

Proof. For the convergence, we first prove that the objective function $\mathbf{P_t}(\mathbf{G}, \mathbf{Y})$ keeps non-increasing when updating $\{\mathbf{G}, \mathbf{Y}\}$. According to lines 3-5 in Alg. 4, we have $\mathbf{P_t}(\mathbf{G}^{i-1}, \mathbf{Y}^{i-1}) \ge \mathbf{P_t}(\mathbf{G}^{i}, \mathbf{Y}^{i-1}) \ge \mathbf{P_t}(\mathbf{G}^{i}, \mathbf{Y}^{i})$, where the first inequality is due to the sub-optimality of UAV trajectory $\mathbf{G}^i$. The second inequality holds because of the sub-optimality of $\mathbf{Y}^i$ by dependent rounding. In addition, the objective function $\mathbf{P_t}(\mathbf{G}, \mathbf{Y})$ is always non-negative. Therefore, the objective function keeps non-increasing after every iteration, which is also finitely lower-bounded by zero. For the complexity, in line 4 of Alg. 4, problem $\mathbf{P'_{t_2}}$ is solved by interior point method, whose computation complexity is $O((IUS)^3)$. Following Lemma 4, the running time of Alg. 3 is $O((IS+U)IUS)$. To summarize, the complexity of Alg. 4 is $O(I^{max}((IUS)^3) + (IS+U)IUS) = O(I^{max}(IUS)^3))$ where I^{max} is the maximum iteration times of Alg. 4, which is polynomial, and this concludes the lemma. □

The overall performance of *OOA* is given by the following theorem.

Theorem 2. OOA *converges in polynomial time.*

Proof. First, it's easy to verify that the complexity of Alg. 2 is $O((US)^2)$, since the **while** loop in lines 2-12 of Alg. 2 terminates in $O(|V|) = O(US)$ iterations. Therefore, the complexity of Alg. 1 is $O((US)^2 + T * ((US)^2 + I^{max}(IUS)^3))) = O(TI^{max}(IUS)^3)$. Since the convergence of Alg. 4 is proofed in Lemma. 5, we obtain that *OOA* converges in polynomial time. □

5 Performance Evaluation

5.1 Evaluation Setup

Parameter Settings. We simulate a UAV-assisted MEC network running for $T = 100$ time slots (a time slot is 20 seconds), with $U \in [2, 10]$ UAVs and $I \in [12, 48]$ UEs. UAVs are randomly scattered in a square area of 200×200 m^2. For the coordinates of UEs, we use the EUA dataset [37], which contains locations of 125 base stations and 816 mobile users in Melbourne central business district area. We choose a base station whose coverage radius is 200 m, and then randomly choose I users in the coverage of the base station as the UEs in our simulation. The network provides $S = 20$ types of services for UEs. The storage capacity of each UAV is $W_u = 3$, while the storage capacity required by service s c_s is within $[0.5, 1]$. Following [15], [28], we use Zipf distribution with exponent value 0.6 as the population of services, and the preferences of UEs $P_{i,s}$ is derived from the population of services with a deviation range from $[-0.1, 0.1]$. The size of each task is $d_{i,s}^t \in [100, 1000]$ KB. The workload of task requiring service s is $\mu_s = [10^6, 10^7]$ cycles (per bytes). Following the similar setting in [23], [24], [33], the parameters of UAVs are set as follows: the computation capacity of UAV for each task $f_u = 1$ GHz, the maximum number of tasks UAV can process at one time slot $N_u^{max} \in [5, 10]$, the maximum flying distance at one slot $D_u^{max} = 50$ m, the maximum covering radius of UAV $R_u^{max} = 200$ m, the fixed altitude $Z_u = 100$ m, the spectrum bandwidth of each communication channel is $B = 1$ MHz. As for energy budget and consumption, according to the properties of DJI Mavic 2 Pro [38], the energy budget of UAV is $E_u^{max} = 4$ Wh[3]. The transmission power of each UAV is set to $p_u = 0.1$ w. The unit flying energy consumption is $\omega_u = 6$ J/m.

Baselines. We compare *OOA* with three algorithms.

- **RANDOM.** Service caching and UAV trajectories are made randomly, and UEs will offload as many tasks as possible to UAVs as long as all the constrains are satisfied.
- **DELAY.** DELAY makes all the decisions to minimize the delay just like *OOA*, without considering saving energy for the future (*i.e.*, $\lambda_u^t = 0, \forall u, \forall t$).
- **TJSO [24].** In TJSO, caching decisions, UAV trajectories and task offloading are jointly optimized periodically. TJSO does not consider strictly satisfying the energy constraint.

5.2 Evaluation Results

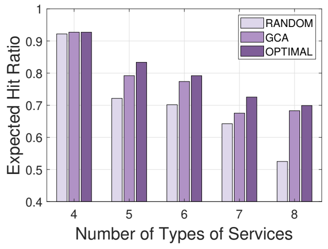


Fig. 3: Expected hit ratio under different numbers of types of services.

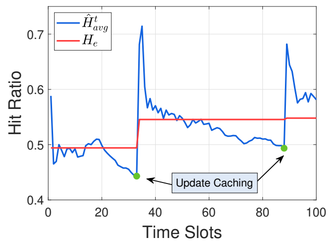


Fig. 4: Average hit ratio of each time slot.

Hit Ratio. Fig. 3 shows the expected hit ratio achieved by different caching algorithms under different numbers of types of services. The expected hit ratio decreases with the increase in the number of types of services because tasks generated by UEs are more diverse while the storage space of UAV remains the same. The RANDOM algorithm randomly caches services in UAVs and performs worst on the expected hit ratio. The OPTIMAL algorithm gives the optimal solution to the caching problem (**P1**), however, it runs for dozens of minutes. Compared with these two algorithms, *GCA* achieves a near-optimal expected hit ratio within one second. Fig. 4 shows the average hit ratio $\hat{H}_{avg}^t$ achieved by *OOA* and corresponding expected hit ratio H_e, where the hit ratio tolerance κ is set to 0.05. It's seen that the gap between $\hat{H}_{avg}^t$ and H_e is within the tolerance in most of the time slots. Once $H_e - \hat{H}_{avg}^{t-1} > \kappa$, caching decisions is updated. Fig. 5 shows the expected hit ratio achieved by different caching algorithms under different storage space. The expected hit ratio increases with the increase in the storage space since larger storage space allows UAVs to cache more services. Similar to Fig. 3, among three algorithms, *GCA* can always achieve a near-optimal solution, and compared with the running time of the OPTIMAL algorithm, which takes several hours, *GCA* only takes less than 1 second.

3. The battery capacity of DJI Mavic 2 Pro is 59.29 Wh, most of which is used for hovering.

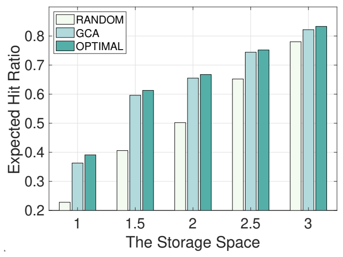


Fig. 5: Expected hit ratio with different storage space W_u.

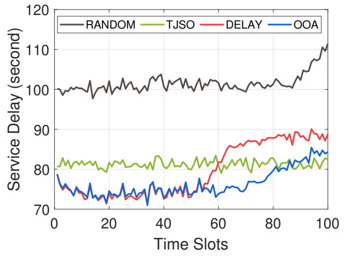


Fig. 6: Service delay of each time slot.

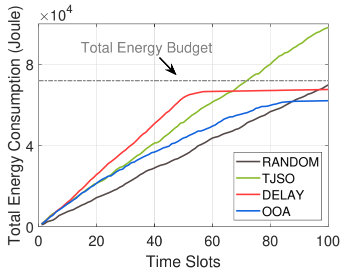


Fig. 7: Cumulative energy consumption.

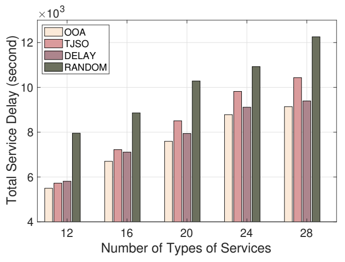


Fig. 8: Service delay with different numbers of types of services S

Service Delay/Energy Consumption. Fig. 6 and Fig. 7 show the service delay of each time slot and UAVs' cumulative energy consumption, respectively. Compared with other three algorithms, *OOA* achieves the lowest service delay under strict energy constraints. The service delay of TJSO is 8% higher than that of *OOA* in the first 80 time slots, and lower than that of *OOA* in the last 20 time slots. However, TJSO ignores the energy constraint and causes the highest energy consumption among four algorithms. DELAY algorithm achieves the lowest service delay at first. Nevertheless, due to the abuse of energy, most of the energy budget is used in the first 60 time slots, causing the service delay of DELAY significantly increases in the last 40 time slots. The RANDOM algorithm performs worst in service delay, which is 33% higher than that of *OOA*. In the last 10 time slots, because *OOA* strictly follows the energy budget constraint, and the remaining energy budgets of most UAVs currently are in short supply, *OOA* no longer changes the positions of UAVs. Since the flight energy consumption occupies much of the previous energy consumption, and many tasks are offloaded to the remote cloud, the energy consumption of *OOA* hardly increases at this time.

Effect of Service Types. Fig. 8 shows the total service delay under different numbers of types of services after 100 time slots. We can see that *OOA* always achieves the lowest service delay with different numbers of service types. The service delay increases as the number of service types increases, since a larger number of types of service makes it difficult for UAVs to cache the services corresponding to user tasks. In addition, when the number of service types is small, the service delay achieved by TJSO is close to that of *OOA*. As the number of service types increases, the gap between OOA and TJSO gradually widens, which shows that it is for TJSO's caching strategy to adapt to the diverse service types.

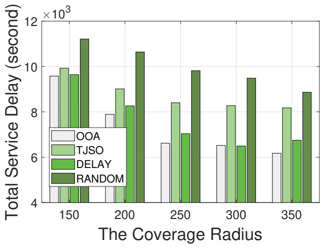


Fig. 9: Service delay with different coverage radius R_u.

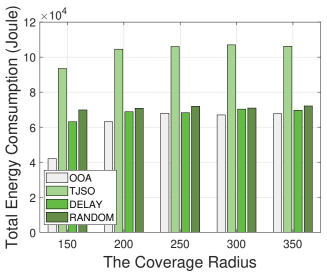


Fig. 10: Energy consumption with different R_u.

Effect of UAV's Coverage Radius. Fig. 9 and Fig. 10 show the total service delay and energy consumption under different coverage radius of UAV after 100 time slots. It can be observed that the service delay decreases as the coverage radius of UAV increases, since a larger coverage radius of UAVs means UAVs are more likely to receive offloaded tasks from UEs. The service delay of *OOA* hardly decreases when the coverage radius is larger than 250 m because a coverage radius of 250 m is enough for UAVs to cover most of the UEs in our simulation.

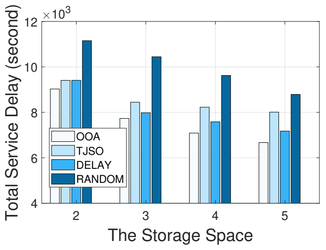


Fig. 11: Service delay with different storage space W_u.

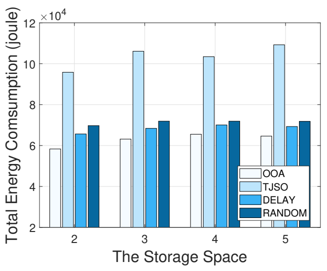


Fig. 12: Energy consumption with different storage space W_u.

Effect of UAV's Storage Space. Fig. 11 and Fig. 12 show the total service delay and energy consumption under different storage space of UAV after 100 time slots. It can be observed that the service delay decreases as the storage space of UAV increases because a larger storage space of UAV means UAVs are more likely to have the services that UEs require. As the storage space increases, the gap between the delay of *OOA* and TJSO becomes larger. That's because a larger storage space will amplify the benefit of *OOA*'s dynamic caching algorithm.

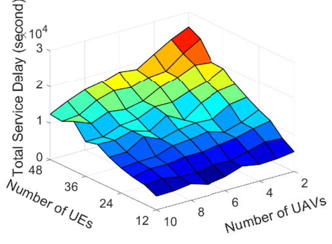


Fig. 13: Service delay with different numbers of UAVs and UEs.

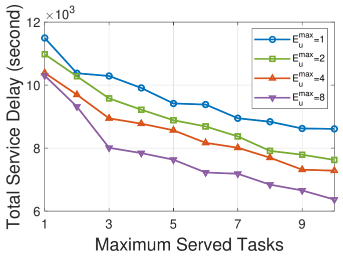


Fig. 14: Service delay with different energy budgets E_u^{max} and maximum served tasks N_u^{max}.

Effect of the Number of UAVs and UEs. Fig. 13 show the total service delay under different numbers of UAV and UEs after 100 time slots. With a fixed number of UEs, the increase in the number of UAVs leads to the decrease in the service delay because more tasks can be offloaded to the UAVs. Notice that when the number of UEs is small, the increase in the number of UAVs can only slightly reduce the service delay, indicating that two UAVs have processed most of the tasks from UEs on this occasion.

Effect of UAV's Energy Budget and computing capacity. Fig. 14 show the total service delay under different energy budgets E_u^{max} and maximum served tasks N_u^{max} of UAVs after 100 time slots. As shown in the figure, the service delay decreases with the increase of UAV's energy budget and maximum served tasks, since more tasks can be processed by UAVs. The service delay of different energy budgets is similar when N_u^{max} is small. This is because in this case, the bottleneck of the delay is the computing capacity of the UAV.

6 Conclusion

In this paper, we study the joint service caching and task offloading problem for UAV-assisted MEC Networks. Different from existing work, we consider a practical scenario where the caching decision is updated dynamically according to UE's preference on services. We further consider the limited energy capacity of UAVs, and jointly optimize UAV trajectory and task offloading based on the caching decision. We propose an online algorithm, *OOA*, that dynamically updates services caching and determines task offloading at each time slot to minimize the overall service delay. *OOA* employs a greedy algorithm to greedily make caching decisions to maximize the sum of the hit ratio, and an iterative algorithm to alternately determine UAV trajectory and task offloading. Both the theoretical analysis and large-scale simulations verify the performance of *OOA*. Simulation results show that *OOA* can reduce the service delay by up to 33%, compared with three benchmarks.

References

[1] J. Cao, X. Liu, X. Su, S. Tarkoma, and P. Hui, "Context-aware augmented reality with 5g edge," in *Proc. of IEEE GLOBECOM*. IEEE, 2021, pp. 1–6.
[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
[3] B. Li, Z. Fei, and Y. Zhang, "Uav communications for 5g and beyond: Recent advances and future trends," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2018.
[4] Xinhua, *China deploys UAV for telecom restoration in rain-hit Henan*, 2021, http://en.people.cn/n3/2021/0723/c90000-9875913.html.
[5] A. Malik, J. Kim, K. S. Kim, and W.-Y. Shin, "A personalized preference learning framework for caching in mobile networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2124–2139, 2020.
[6] X. Zhang, H. Li, J. Wang, Y. Guo, Q. Pei, P. Li, and M. Pan, "Data-driven caching with users' content preference privacy in information-centric networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5744–5753, 2021.
[7] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. of IEEE INFOCOM*. IEEE, 2020, pp. 2066–2075.

[8] V. Farhadi, F. Mehmeti, T. He, T. F. La Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 779–792, 2021.

[9] D.-T. Do, A.-T. Le, Y. Liu, and A. Jamalipour, "User grouping and energy harvesting in uav-noma system with af/df relaying," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11 855–11 868, 2021.

[10] D.-H. Tran, V.-D. Nguyen, S. Chatzinotas, T. X. Vu, and B. Ottersten, "Uav relay-assisted emergency communications in iot networks: Resource allocation and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1621–1637, 2021.

[11] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and X. Tao, "Distributed online learning of cooperative caching in edge cloud," *IEEE Transactions on Mobile Computing*, vol. 20, no. 8, pp. 2550–2562, 2020.

[12] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2020.

[13] Y. Han, L. Ai, R. Wang, J. Wu, D. Liu, and H. Ren, "Cache placement optimization in mobile edge computing networks with unaware environment—an extended multi-armed bandit approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 8119–8133, 2021.

[14] S. Gu, X. Sun, Z. Yang, T. Huang, W. Xiang, and K. Yu, "Energy-aware coded caching strategy design with resource optimization for satellite-uav-vehicle-integrated networks," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5799–5811, 2021.

[15] R. Zhang, R. Lu, X. Cheng, N. Wang, and L. Yang, "A uav-enabled data dissemination protocol with proactive caching and file sharing in v2x networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3930–3942, 2021.

[16] L. Li, M. Wang, K. Xue, Q. Cheng, D. Wang, W. Chen, M. Pan, and Z. Han, "Delay optimization in multi-uav edge caching networks: a robust mean field game," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 808–819, 2020.

[17] K. Wang, X. Zhang, L. Duan, and J. Tie, "Multi-uav cooperative trajectory for servicing dynamic demands and charging battery," *IEEE Transactions on Mobile Computing*, 2021.

[18] Z. Ning, P. Dong, M. Wen, X. Wang, L. Guo, R. Y. Kwok, and H. V. Poor, "5g-enabled uav-to-community offloading: Joint trajectory design and task scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3306–3320, 2021.

[19] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang, and X. Lang, "Learning-based computation offloading approaches in uavs-assisted edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 928–944, 2021.

[20] C. Sun, W. Ni, and X. Wang, "Joint computation offloading and trajectory planning for uav-assisted edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5343–5358, 2021.

[21] Q. Tang, L. Liu, C. Jin, J. Wang, Z. Liao, and Y. Luo, "An uav-assisted mobile edge computing offloading strategy for minimizing energy consumption," *Computer Networks*, vol. 207, p. 108857, 2022.

[22] M. Zhang, E.-H. Mohammed, and S. X. Ng, "Intelligent caching in uav-aided networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 739–752, 2021.

[23] Y. Qu, H. Dai, H. Wang, C. Dong, F. Wu, S. Guo, and Q. Wu, "Service provisioning for uav-enabled mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3287–3305, 2021.

[24] R. Zhou, X. Wu, H. Tan, and R. Zhang, "Two time-scale joint service caching and task offloading for uav-assisted mobile edge computing," in *Proc. of IEEE INFOCOM*. IEEE, 2022, pp. 1189–1198.

[25] J. Luo, J. Song, F.-C. Zheng, L. Gao, and T. Wang, "User-centric uav deployment and content placement in cache-enabled multi-uav networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5656–5660, 2022.

[26] J. Ji, K. Zhu, D. Niyato, and R. Wang, "Joint cache placement, flight trajectory, and transmission power optimization for multi-uav assisted wireless networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 8, pp. 5389–5403, 2020.

[27] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Joint resource, deployment, and caching optimization for ar applications in dynamic uav noma networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3409–3422, 2021.

[28] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal uav caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 12, pp. 2783–2797, 2020.

[29] Y. Wu, S. Yao, Y. Yang, Z. Hu, and C.-X. Wang, "Semigradient-based cooperative caching algorithm for mobile social networks," in *Proc. of IEEE GLOBECOM*. IEEE, 2016, pp. 1–6.

[30] B. Chen and C. Yang, "Caching policy for cache-enabled d2d communications by learning user preference," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6586–6601, 2018.

[31] M. J. Magazine and M.-S. Chern, "A note on approximation schemes for multidimensional knapsack problems," *Mathematics of Operations Research*, vol. 9, no. 2, pp. 244–247, 1984.

[32] A. Kulik and H. Shachnai, "There is no eptas for two-dimensional knapsack," *Information Processing Letters*, vol. 110, no. 16, pp. 707–710, 2010.

[33] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for uav-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, 2021.

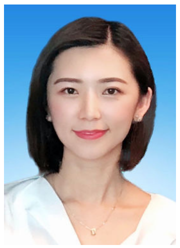
[34] M. Grant and S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*, 2014, http://cvxr.com/cvx.

[35] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," 2005.

[36] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

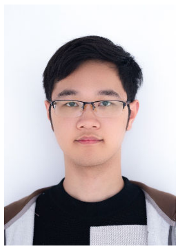
[37] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. of ICSOC*. Springer, 2018, pp. 230–245.

[38] DJI, *DJI Mavic 2 Pro*, 2022, https://www.dji.com/cn/mavic-2-enterprise-advanced/specs.



Ruiting Zhou is an Associate Professor in the School of Computer Science Engineering at Southeast University. She received her Ph.D. degree in 2018 from the Department of Computer Science, University of Calgary, Canada. Her research interests include cloud computing, machine learning and mobile network optimization. She has published research papers in top-tier computer science conferences and journals, including IEEE INFOCOM, ACM MOBIHOC, IEEE/ACM TON, IEEE JSAC, IEEE TMC. She serves as the TPC chair for INFOCOM workshop-ICCN 2019-2023. She also serves as a reviewer for international conferences and journals such us IEEE ICDCS, IEEE/ACM IWQoS, IEEE SECON, IEEE JSAC, IEEE TON, IEEE TMC, IEEE TCC

Yifeng Huang received the B.E. degree at School of Computer Science, Wuhan University, China, in 2021. He is currently pursuing the M.S. degree in the School of Cyber Science and Engineering at Wuhan University. His research interests include edge computing, UAV-enabled wireless networks, and online scheduling.



Yufeng Wang received the B.E. degree in Information Security from Wuhan University, China, in 2023. Now he is pursuing his master's degree in Institute for Network Science and Cyberspace at Tsinghua University. His research interests include UAV-enabled wireless networks, network optimization and satellite networking.

Lei Jiao received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is currently an assistant professor at the Department of Computer Science, University of Oregon, USA. Previously he worked as a member of technical staff at Nokia Bell Labs in Dublin, Ireland and as a researcher at IBM Research in Beijing, China. He is interested in the mathematics of optimization, control, learning, and economics applied to computer and telecommunication systems, networks, and services. He publishes papers in journals such as JSAC, ToN, TPDS, TMC, and TDSC, and in conferences such as INFOCOM, MOBIHOC, ICNP, ICDCS, SECON, and IPDPS. He is an NSF CAREER awardee. He also received Best Paper Awards of IEEE LANMAN 2013 and IEEE CNS 2019. He was on the program committees of many conferences, including INFOCOM, MOBIHOC, ICDCS, IWQoS, and WWW, and was also the program chair of multiple workshops with INFOCOM and ICDCS.

Haisheng Tan received his B.E. degree in Software Engineering and B.S. degree in Management both from University of Science and Technology of China (USTC) with the highest honor. Then, he got his Ph.D. degree in computer science at the University of Hong Kong (HKU). He is currently a professor at USTC. His research interests lie primarily in Networking Algorithm Design and System Implementation, where he has published over 90 papers in prestigious journals and conferences. He recently received the awards of ACM China Rising Star (Hefei Chapter), the Distinguished TPC Member of INFOCOM 2019, the Best Paper Award in WASA'19, CWSN'20, PDCAT'20 and ICPADS'21. He is a senior member of IEEE.

Renli Zhang received a B.E. degree in Information Security from Wuhan University, China, in 2020. He is currently pursuing the M.S. degree in the School of Cyber Science and Engineering at Wuhan University. His research interests include UAV-enabled wireless networks, network optimization, and online scheduling.

Libing Wu received the Ph.D. degree from Wuhan University, China, in 2006. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He was a visiting scholar with the Advanced Networking Lab, University of Kentucky, USA, in 2011. He was a senior visiting fellow with State University of New York, Buffalo, USA, in 2017. His research interests include network security, Internet of Things, machine learning and data security.