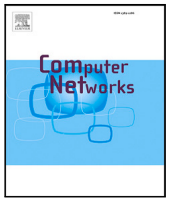
Computer Networks 218 (2022) 109405

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



MUTAA: An online trajectory optimization and task scheduling for UAV-aided edge computing☆

Weidu Ye, Junzhou Luo *, Wenjia Wu, Feng Shan, Ming Yang

School of Computer Science and Engineering, Southeast University, Nanjing, China

ARTICLE INFO

Keywords:
UAV-aided edge computing
Trajectory optimization
Task scheduling
Online algorithm

ABSTRACT

A novel UAV-aided edge computing system is proposed in this work, where UAV-aided edge nodes are dispatched to provide communication and computation assistance for completing tasks generated by ground clients (GCs). We formulate a trajectory design and task allocation problem (TDTAP), aiming at maximizing the sum of completed tasks of GCs by optimizing the proper trajectory for each UAV and scheduling tasks from each GC. It is impossible to solve the TDTAP problem directly in polynomial time since UAVs lack all GCs' information, e.g., position and amount of tasks. To this end, we put forward an online iterative algorithm named Maximum UAV trajectory and Task Allocation Algorithm (MUTAA) to solve the TDTAP problem by jointly optimizing UAVs' trajectory and GCs' task scheduling. Unlike existing algorithms, MUTAA can make real-time decisions for each UAV without acquiring information from all GCs in advance. During each iteration, MUTAA consists of two sub-algorithms: (1) trajectory design algorithm TDA and (2) task allocation strategy TAS. Specifically, the preschedule step is used in TDA to find the proper trajectory for UAVs, and a competitive online algorithm, TAS, is proposed to schedule GCs' tasks. Theoretical analysis proves that TAS is $e/(e-1)$-competitive, that is, it processes $(e-1)/e$ (approximately 63%) tasks when compared with the optimal offline solution. Experimental results demonstrate that MUTAA completes 83% data on an average of the optimal offline solution, illustrating that the proposed algorithm MUTAA can be widely used in time-sensitive scenarios.

1. Introduction

The increasing number of ground clients (GCs), e.g., wearable devices, smartphones, and tablet computers, has produced enormous computation-intensive tasks, such as video calls, traffic surveillance, and online games [1]. However, it is generally hard for a GC to complete the enormous computing tasks in time by itself due to its limited computing resources on board. Although offloading some tasks to a base station (BS) can alleviate this problem to some degree, it might cause congestion and unbearable network delay. By deploying nearby edge nodes with richer computation resources, the GCs' burden can be sharply reduced by offloading tasks to edge nodes [2].

Due to high mobility and flexibility, unmanned aerial vehicles (UAVs) have been widely adopted in edge computing (EC) [3]. UAV-aided edge computing has many advantages compared with terrestrial edge computing. First, UAVs can improve the communication services for GCs, for they can fly close to GCs and provide line-of-sight (LOS) communication, which improves the transmission rate between UAVs and GCs. Second, unlike terrestrial edge computing, UAVs can offer timely services due to their mobility to move quickly, which can be applied in emergencies.

Currently, most works [4–16] in UAV-aided edge computing focus on finding out an appropriate strategy to minimize the energy consumption [4–10], computation and communication latency [11–13], or maximize the tasks' completion amount [14–16] of GCs. In these works, sophisticated UAV-aided edge computing models are formulated and transferred as non-convex problems. These problems are solved by offline algorithms by mathematic tools such as CVX and Gurobi and achieve near-optimal results. However, using maths tools takes much computation time, and UAVs can only obtain information from GCs within their communication range, leading to the result that UAVs cannot make real-time decisions. Therefore, an online UAV-aided edge computing strategy that consists of a BS, UAVs, and GCs is needed with a more significant task completion amount and less execution time to meet the demand of GCs.

☆ This work was partially supported by the National Natural Science Foundation of China (Nos.62232004, 62132009,62072102, 62072101, and 62072103); Jiangsu Provincial Key Laboratory of Network and Information Security (No. BM2003201); the Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China (No. 93K-9).

* Corresponding author.
E-mail address: jluo@seu.edu.cn (J. Luo).

https://doi.org/10.1016/j.comnet.2022.109405
Received 15 April 2022; Received in revised form 25 September 2022; Accepted 2 October 2022
Available online 12 October 2022

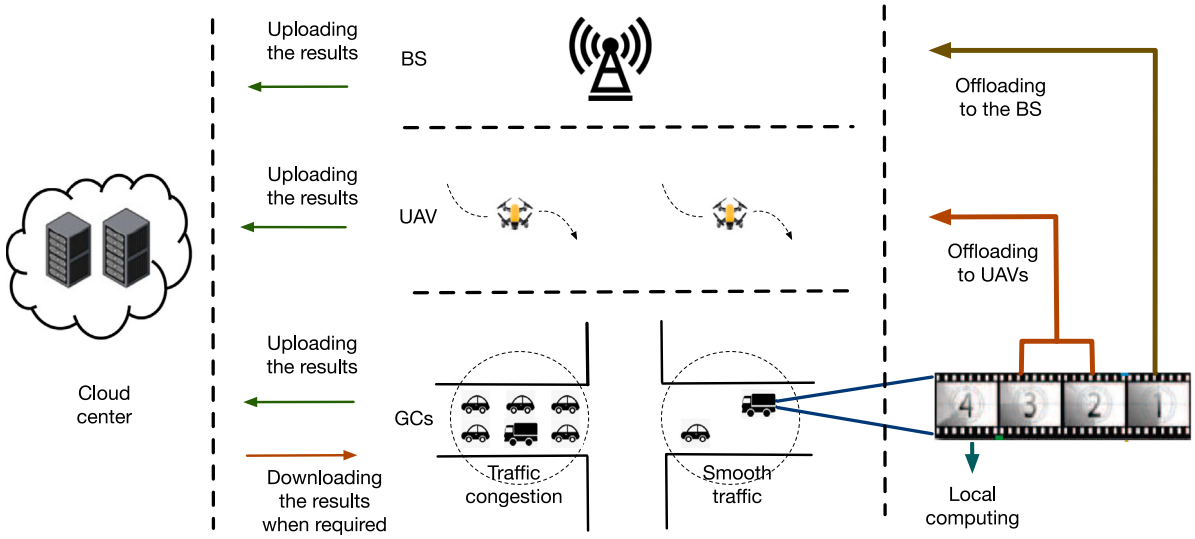


Fig. 1. An application scenario of UAV-aided edge computing system. Several GCs, such as tachograph devices, are set on vehicles to record real-time traffic video. Using video processing technology, each device can obtain information on traffic conditions such as traffic congestion, identify the traffic condition and upload the results to keep other vehicles from traffic congestion, traffic accident, and traffic signal from the recorded video. Due to their limited computation capability, tachograph devices cannot complete all video-processing tasks alone. They divide the video into several small images by frames and offload some image-processing tasks to the BS and UAV-aided edge nodes to compute. The computation results (usually much smaller than computation tasks) will be uploaded to the remote cloud center. When other GCs require the traffic information, they can download the results from the remote cloud center. In this work, we aim to maximize the total completed tasks from GCs by considering the trajectory of each UAV and task-scheduling strategy for each GC.

One application scenario of the UAV-aided edge computing system is shown in Fig. 1, which contains GCs, several UAV-aided edge nodes, a BS, and a central cloud. Specifically, a group of GCs (such as tachograph devices) is set in vehicles to record the real-time traffic video. Using video processing technology, a GC can acquire traffic information such as traffic congestion, traffic accident, and traffic signal from the recorded video. However, the computation capability of GCs is limited, and they cannot process the video-processing task alone. The video-processing task is then divided into several image-processing tasks by frames, and some of them are offloaded to the BS, or some UAV-aided edge nodes [17]. The processing results will be uploaded to the cloud center for traffic monitoring and dangerous driving behavior detection. Since processing results are much smaller than video-processing tasks, when other GCs require traffic information, they can download the results from the remote cloud center. This work aims to maximize the total processing data generated from GCs by optimizing each UAV's trajectories and scheduling task allocation for each GC.

The readers can capture the following challenges we face. (1) Unlike terrestrial edge computing, UAV-aided edge computing considers the task scheduling strategy among GCs, UAVs, and the BS but also considers trajectory design. In practice, UAVs receive tasks offloaded from GCs while flying simultaneously. Thus, a joint UAV trajectory and task scheduling strategy should be proposed in this work. (2) Most existing works in UAV-aided edge computing systems do not consider the centralized BS. A few works with BS exist, but they solve UAV trajectory and task scheduling problems in offline scenarios, where UAVs must obtain information in advance from GCs, such as their location and residual task amount. UAVs usually lack the information above, and thus an online strategy needs to be implemented in the corresponding scenarios.

Rising to the above challenges, this paper proposes an online algorithm named Maximum UAV Trajectory and Task Allocation Algorithm (MUTAA) to optimize UAVs' trajectories and schedule GCs' tasks. The main contributions are listed as follows.

- This paper presents a novel UAV-aided edge computing scenario, where a BS, UAVs, and GCs cooperatively process tasks generated by GCs. Unlike previous works, UAVs do not know locations and total task amounts of GCs in advance, thus needing real-time decisions. We consider UAVs' trajectories and GCs' task scheduling and formulate a TDTAP problem to maximize the task-processing amount of GCs.
- We propose an online algorithm named MUTAA to solve TDTAP, and MUTAA is consisted of two sub-algorithms to optimize UAVs' trajectories and schedule GCs' tasks, respectively. Specifically, the trajectory design algorithm TDA is first proposed to optimize trajectories of UAVs by using preschedule strategy, which improves the task processing amount of GCs. Then a competitive online algorithm, TAS, is presented to schedule GCs' tasks. We utilize primal–dual technology in TAS and prove that TAS is $\frac{e}{e-1}$-competitive, that is, it completes at least $\frac{e-1}{e}$ (approximately 63%) data as much as the optimal offline solution through theoretical analysis.
- The task-allocation experimental results demonstrate that MUTAA completes an average 83% offloading task amount of the optimal offline policy but takes much less execution time. Besides, the trajectory design experimental results show that UAVs have a shorter flight distance in MUTAA than the compared algorithm.

The rest of this paper is organized as follows. Section 2 introduces related works on UAV-aided edge computing and task allocation strategies in UAV-aided edge computing. Section 3 presents the system model and then formulates the TDTAP problem. Section 4 proposes an online algorithm MUTAA to solve the TDTAP problem. Section 5 conducts experiments to verify the performance of our algorithm. Finally, Section 6 concludes the work.

2. Related works

This section reviews recent related works on UAV-aided edge computing, task dispatching and scheduling strategy, and task dispatching and scheduling strategy in UAV-aided edge computing.

2.1. UAV-aided edge computing

Many works exist concentrating on designing a UAV-aided edge computing system to offload the computing tasks generated by GCs.

Yong et al. [3] investigated many recent works in UAV-aided edge computing and declared that UAV-aided edge computing could process more tasks than traditional terrestrial edge computing. Hu et al. [18] assumed that computing tasks generated from each GC have their deadlines and aimed to minimize the energy consumption of all GCs by optimizing UAV's trajectory. Zhou et al. [19] combined a UAV-enabled wireless communication system with wireless power transfer (WPT) to collect data from ground IoT devices, where the UAV can harvest energy to extend their working time. On this basis, they aimed to maximize the transmission rate of all IoT devices while considering the energy consumption of the UAV.

Some works utilize UAV swarms to assist GCs in processing their tasks cooperatively. Cao et al. [20] novelly developed a UAV-aided edge computing system by utilizing swarm intelligence technology, where several UAVs were dispatched to serve large-scale IoT devices. To minimize the total energy consumption cost by UAVs and IoT devices, an offline strategy is proposed to optimize the trajectory of each UAV and schedule a task-allocation strategy for each IoT device. Jeong et al. [21] mounted computing edge nodes on UAVs and assumed them as mobile cloudlets to process tasks from ground users in emergency scenarios such as earthquakes and fire disasters. A jointly UAV-trajectory and task-allocation algorithm was also proposed in this paper to minimize the total delay of all computing tasks generated by ground users.

Most of these works study edge computing scenarios consisting of UAV-aided edge nodes and GCs. However, the edge computing system should consider the centralized BS in practicality. In this work, a three-layer scenario is constructed consisting of a BS, several UAVs, and GCs to optimize UAVs' trajectories and GCs' task scheduling jointly.

2.2. Task dispatching and scheduling strategy

Task dispatching and scheduling strategy focus on dividing the computation tasks into several sub-tasks and scheduling the sub-tasks to the proper device to minimize the task-completion time. Deng et al. [22] constructed a system contains several wireless APs to process data uploaded from users, and proposed an online AP-user association algorithm to maximize the amount of offloaded data. Meng et al. [23] proposed an online task dispatching algorithm that maximizes the total task processing amount for GCs. Specifically, each GC divides the task into multiple subtasks according to the task type (e.g., video-processing tasks, speech recognition tasks) and uploads their subtasks to different edge nodes for task processing. Tan et al. [24] proposed a system that dispatches and schedules that the job response time is minimized. In their policies, the job is divided into several sub-jobs according to time slots, and each sub-jobs can be severed by the edge server instantly. The processing results are then returned to GCs.

Besides, some video-segmentation works divide a video into several images by frames. Song et al. [25] proposed a video-segmentation model that divided the whole video into several small images by frames. Each frame is processed separately to shorten the total processing time of the video. Liu et al. [26] proposed an online semi-supervised video segmentation approach named GCseg to deal with video-processing tasks in a short time. The GCseg contained a co-segmentation module to divide the video into several small images by frames and then process these images simultaneously. Quellec et al. [27] presented a solution to automatically categorize surgical tasks in real-time during the surgery using video recording. It used video segmentation to divide video into several images and analyzed the current surgery by comparing archived images. The solution could provide valuable recommendations to less experienced surgeons.

2.3. Task dispatching and scheduling strategy in UAV-aided edge computing network

Several works combine task dispatching and scheduling strategies in UAV-aided edge computing [13,28–30]. Most existing works construct a joint UAV trajectory and task allocation problem and use mathematical tools such as Matlab and CVX to solve the problem. A novel UAV-aided edge computing scenario including the UAV and several edge nodes was proposed in work [13] to collaboratively provide computation services for the GCs. They formulated the maximum clique technology to plan a proper trajectory for each UAV and then utilized a task-allocation strategy to offload GCs' tasks. Li et al. [28] studied a UAV-aided mobile edge computing scenario to minimize the task completion time of the UAV by deciding the optimal processing method for tasks from each user. An algorithm using reinforcement learning was proposed to maximize the task processing of all GCs in their work, and experimental results show that their algorithm completed 90% of tasks. Liu et al. [29] mounted the edge node on each UAV to monitor the real-time traffic and proposed an algorithm that minimizes the total required energy by jointly optimizing the CPU frequencies. The offloading amount, transmit power, and UAV's trajectory. A UAV-aided edge computing system was proposed in work [8], where a sophisticated offloading model was established to allocate computing tasks of ground sensors to the UAV or the centralized BS. Zhang et al. [30] equipped a MEC server on a UAV to serve several IoT devices. Each IoT device was assumed to contain a time-critical task to complete, and the author proposed a near-optimal solution to minimize the total latency time of all IoT devices.

However, most of these works considered an offline framework where UAV obtains information from GCs in advance, which did not cater for the situations requiring quick response [13,28,29]. This paper proposes a real-time UAV-aided edge computing system where UAVs and GCs have to make real-time decisions.

3. System model and problem formulation

In this section, we present the system model and formulate the trajectory design and task allocation problem TDTAP, where the task processing amount of all GCs is maximized under the constraint of the UAV's trajectory and computation capability and GCs.

3.1. System model

A UAV-aided edge computing system is established, consisting of m GCs (denoted as $I = \{i_1, i_2, i_3, \dots, i_m\}$), n UAVs (denoted as $J = \{j_1, j_2, j_3, \dots, j_n\}$), and a BS B_b.

We define T as the total time in consideration, and $T = \Delta t \cdot s$, where Δt is the length of each time slot. The value of Δt is small enough that UAVs and GCs are assumed to keep stationary during a single time slot, and s represents the number of time slots during the period T. Symbol K is the set of time slot, denoted as $K = \{k_1, k_2, \dots, k_s\}$.

For each GC $i \in I$, it is assumed to move dynamically and its coordinate in time slot k is defined as $u_{ik} = (s_{ik}, t_{ik}, 0)$, and has a total amount of c_i task required to be completed. For each UAV $j \in J$, its coordinate in time slot k is defined as $q_{jk} = (s_{jk}, t_{jk}, H)$, where H is the flight altitude of UAV. We define v_{max} as the maximum flight speed of UAVs, where the position of each UAV between two adjacent slots cannot exceed v_{max}. Each UAV j contains a communication range R_j and only serves GC i within the communication range R_j. The location of BS $B_b = (s_b, t_b, 0)$ is set in the center of the system, and the computation capability of BS is larger than those of UAVs and GCs.

There are mainly three ways for GC i to process its task.

Table 1
Key parameter definitions.

Parameter	Definition
I	Set of GCs
J	Set of UAVs
K	Set of time slots
i	Single GC
j	Single UAV
b	BS
k	Single time slot
u_{ik}	Location of GC i in slot k
q_{jk}	Location of UAV j in slot k
c_{ik}^{loc}	Local computation rate for GC i in slot k
c_{ik}^{BS}	Transmission rate for GC i to the BS in slot k
c_{ijk}^{UAV}	Transmission rate for GC i to UAV j in slot k
x_{ijk}^{UAV}	Time portion for UAV j to serve GC i in slot k
y_{ibk}^{BS}	Time portion for BS b to serve GC i in slot k
z_{ik}^{loc}	Time portion for GC i to compute locally in slot k
D_{ik}^{loc}	Local computation amount of GC i in slot k
D_{ibk}^{BS}	BS-offloading amount of GC i in slot k
D_{ijk}^{UAV}	UAV-offloading amount from GC i to UAV j in slot k
R_j	Communication range for UAV j
c_i	Total task amount for GC i
v_{max}	Maximum flight distance of each UAV
dis_{min}	Safety flight distance between neighbor UAVs

3.1.1. Offloading to UAV

The channel gain h_{ijk} between UAV j and GC i in slot k is defined as:

$$h_{ijk} = \frac{\beta_0}{\|q_{jk}-u_{ik}\|_2^2+H^2} \tag{1}$$

where β_0 is the received power in reference distance (e.g., 1 m) between transmitter and receiver. $\|.\|$ denotes the L2 norm.

The transmission rate between GC i and UAV j is defined as c_{ijk}^{UAV} when GC i offloads tasks to UAV j, whose value is:

$$c_{ijk}^{UAV} = B \cdot log_2(1 + \frac{P \cdot h_{ijk}}{\sigma^2}) \tag{2}$$

where P is the constant transmit power of GC i. σ^2 is the Gaussian noise in the environment.

We define a time portion set x representing the portion of time that GCs offloaded the task to UAV, and variable $x_{ijk}^{UAV} \in x$ is ranging from $[0, 1]$, denoting that when GC i offloads the task to UAV j in slot k.

Hence the UAV-offloading amount D_{ijk}^{UAV} from GC i to UAV j in slot k is:

$$D_{ijk}^{UAV} = x_{ijk}^{UAV} \cdot c_{ijk}^{UAV} \cdot \Delta t \tag{3}$$

We do not consider the computation capability of UAVs, that is because, the computation capability of a UAV is usually larger than its transmission rate. For instance, NVIDIA has developed the Jetson TX2 chip. This chip can work out an average of 225 figures per second [17] and has been widely used in UAVs. Besides, the transmission rate of UAV is more valuable, for UAVs are usually dispatched in locations that are far from the BS but with plenty of GCs around. On this basis, we mainly consider the transmission rate of UAVs in this work, i.e., D_{ijk}^{UAV} does not contain the computation capability of UAV.

3.1.2. Offloading to BS

Similar to previous subsection, the channel gain h_{ibk} between BS b and GC i in slot k is defined as:

$$h_{ibk} = \frac{\beta_0}{\|B_b-u_{ik}\|_2^2+H^2} \tag{4}$$

The transmission rate between GC i and BS b is defined as c_{ibk}^{BS}, whose value is:

$$c_{ibk}^{BS} = B \cdot log_2(1 + \frac{P \cdot h_{ibk}}{\sigma^2}) \tag{5}$$

We define a time portion set y representing the portion of time that GCs offloaded the task to the BS, and variable $y_{ibk}^{BS} \in y$ is ranging from $[0, 1]$, denoting the time portion that GC i offloads the task to the BS in slot k.

The BS-offloading amount D_{ibk}^{BS} of GC i in slot k can be defined as:

$$D_{ibk}^{BS} = y_{ibk}^{BS} \cdot c_{ibk}^{BS} \cdot \Delta t \tag{6}$$

We ignore the computation time of BS, i.e., D_{ibk}^{BS} does not take the computation capability into consideration. The reason is that the computation capability of BS is much larger than that of UAVs and GCs, causing the result that computation time of BS is very short. Therefore, the processing time of BS is mainly affected by transmission time between the BS and GCs.

3.1.3. Local computing

The computation capability of GC $i \in I$ is defined as c_{ik}^{loc}, when GC i processes tasks by itself. Similar to previous subsections, we define a time portion set z representing the portion of time that GCs that process tasks by themselves, and variable $z_{ik}^{loc} \in z$ ranges from $[0, 1]$, denoting the time portion that GC i process tasks in local at slot k.

Thus, the local computation amount D_{ik}^{loc} of GC i in slot k is defined as:

$$D_{ik}^{loc} = z_{ik}^{loc} \cdot c_{ik}^{loc} \cdot \Delta t \tag{7}$$

where c_{ik}^{loc} is the computation capability for GCs to compute the task locally.

Different from UAVs and the BS, each GC has to pre-process the recorded videos before offloading them to UAVs and the BS. For instance, in our scenario, each GC has to divide a recorded video into several small pieces by frames, this procedure may cost much time [26,27]. Therefore, when GCs offload a task to UAVs or the BS, the communication module of GCs uploads their tasks to UAVs or the BS, and the computation module has to divide the upcoming video into several small videos. To this end, GCs do not have time to compute locally while offloading tasks to UAVs or the BS.

Overall, we conclude all the parameters in Table 1.

3.2. Problem formulation

In this section, constraints on GCs, UAVs, and the BS in UAV-aided edge computing systems are first introduced, and then the TDTAP problem is proposed.

Definition 1 (*Slot Allocation Constraints*)**.** The slot allocation constraints explain the relationship of time slot allocation among UAVs, the BS, and GCs, which limit the computation time for each device.

First, for each UAV, the constraint is written in Eq. (C1):

$$\sum_{i=1}^{m} x_{ijk}^{UAV} \leq 1 \tag{C1}$$

where for each UAV j at time slot $k \in K$, the total time portion allocated by UAV j to process all GCs' tasks cannot exceed the length of a single slot.

Second, for each BS, the constraint is written in Eq. (C2):

$$\sum_{i=1}^{m} y_{ibk}^{BS} \leq 1 \tag{C2}$$

where at each time slot $k \in K$ for BS b, the total time portion allocated by BS b to process all GCs' tasks cannot exceed the length of a single time slot.

Finally, for each GC i, according to the assumption that GCs cannot compute and offload the tasks simultaneously, the constraint is written in Eq. (C3)

$$\sum_{j=1}^{n} x_{ijk}^{UAV} + y_{ibk}^{BS} + z_{ik}^{loc} \leq 1 \tag{C3}$$

where at each time slot $k \in K$ for each GC i, the total time portion allocated from all UAVs, BS b, and GC itself to process GC i's tasks cannot exceed the length of a single slot.

Definition 2 (*UAV's Trajectory Constraints*)**.** The UAV's trajectory constraints illustrate the flight state of each UAV, which limit the maximum flight speed, communication range, and safety distance.

First, at each time slot Δt, a maximum speed of each UAV j cannot surpass v_{max}, shown in (C4):

$$\|q_{j(k+1)} - q_{jk}\| \le v_{max} \cdot \Delta t, \tag{C4}$$

where q_{jk} represents the position of UAV j at time slot k.

Second, at each time slot $k \in K$, if GC i offloads tasks to UAV j, the distance between GC i and UAV j cannot exceed the communication range of UAV R_j, shown in (C5):

$$x_{ijk}^{UAV} \cdot \|q_{jk} - u_{ik}\| \le R_j, \tag{C5}$$

Third, at each time slot $k \in K$, each UAV j should keep a distance from other UAVs for safety. We assume $\{a, b\} \in J$ are two different UAVs, shown in (C6):

$$\|q_{ak} - q_{bk}\| \ge dis_{min}, \tag{C6}$$

Definition 3 (*Maximum Amount Constraints*)**.** The Maximum amount constraints describe the maximum task processing amount of GC $i \in I$, which restricts the total task processing amount of each G, that is

$$\sum_{j=1}^{n}\sum_{k=1}^{s} D_{ijk}^{UAV} + \sum_{k=1}^{s} D_{ibk}^{BS} + \sum_{k=1}^{s} D_{ik}^{loc} \le c_i. \tag{C7}$$

where the total amount of tasks computed by UAVs, the BS, and GCs cannot exceed the maximum task amount of GC i.

According to above definitions, the total task processing amount D_{sum} of GC i at period T is

$$\begin{aligned} D_{sum} = &\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{s} D_{ijk}^{UAV} \\ &+ \sum_{i=1}^{m}\sum_{k=1}^{s} D_{ibk}^{BS} \\ &+ \sum_{i=1}^{m}\sum_{k=1}^{s} D_{ik}^{loc} \end{aligned} \tag{8}$$

We define the trajectory design and task allocation problem (TDTAP) as follows:

Definition 4 (*TDTAP Problem*)**.** Given I GCs, J UAV-aided edge computing nodes, and a BS, the TDTAP problem is proposed for optimizing the trajectory of each UAV and scheduling task allocation of each GC to maximize the total task processing amount of GCs in period T (Eq. (C5)). Meanwhile, the problem needs to satisfy the *slot allocation constraints* Eqs. (C1)–(C3), *UAV trajectory constraints* Eqs. (C4)–(C6), and *maximum amount constraints* Eq. (C7).

The TDTAP problem is consequently written in **P1**:

$$(P1): \max_{x,y,z,c_{ijk}^{UAV},q} D_{sum} \tag{9}$$

$$\text{s.t.} \quad (C1) - (C7) \tag{10}$$

Mathematic tools cannot solve the problem above since trajectory variable q and transmission rate c_{ijk}^{UAV} are unknown. Thus, we propose an online algorithm to solve the problem in the following section.

4. Online algorithms to solve TDTAP

In this section, we propound an online algorithm, MUTAA (Section 4.1), to solve the TDTAP problem by jointly optimizing UAVs' trajectory and scheduling GCs' tasks. MUTAA is an iterative algorithm with two steps in each iteration: Section 4.2 optimizes UAV's trajectory by algorithm TDA and Section 4.3 schedules GCs' tasks by algorithm TAS. MUTAA iterates in all slots in set K, where algorithm TDA and TAS run alternatively to achieve the final results.

4.1. Online algorithm MUTAA

Different from algorithms that design UAVs' trajectories at each time slot, MUTAA uses preschedule steps to find reasonable locations of UAVs (satisfying constraint (C4)∼(C6)) in the next *step* ($step > 1$) time slots for each iteration.

The advantage of using the preschedule step is that it can improve the task process amount of GCs while shortening the total flight distance of UAVs simultaneously. Since the preschedule step considers several time slots instead of a single slot, it could shorten the total flight length for UAVs by preventing them from flying back and forth.

The pseudo-code for MUTAA is shown in *Algorithm* 1.

Algorithm 1: MUTAA (U, R, K)

Input: Initial location I_i of GCs i
Communication range R_j of UAV j
Total task amount c_i of GC i
Prescheduled steps for each iteration *step*
Output: Trajectory for each UAV q
Task allocation variable x, y, z
Total task processing amount $data$

```
Initialize trajectory q and task allocation variables x, y, z
task = c;
while k · Δt ≤ T do
    Update GCs' location u_ik
    Using algorithm 2 to optimize UAVs' trajectories q
    Using algorithm 3 to calculate the total task processing
      amount data and corresponding variables x, y, z.
    k = k + step
end
```

Algorithm 1 iterates each time slot $k \in K$, collects the current information (including UAVs' current locations and residual tasks' amount for GCs), and utilizes the trajectory design algorithm (Algorithm 2) to optimize the trajectory of each UAV. Then, a task allocation algorithm (Algorithm 3) is proposed to offload GCs' tasks to UAV-aided edge nodes or the centralized BS. The MUTAA executes Algorithm 2 and Algorithm 3 iteratively until $k = T/\Delta t$.

4.2. Trajectory design algorithm

In this subsection, we propose a trajectory design algorithm named TDA for each UAV j between time slot k and $k + step - 1$.

Detailed pseudo-code is given in *Algorithm 2*:

Before introducing *Algorithm 2*, we first proposed a concept of residual transmission rate, defined as follows.

Definition 5 (*Residual Transmission Rate*)**.** The residual transmission rate is a parameter that assists each UAV in finding proper GCs with maximum transmission rate and residual GCs' tasks, denoted as $C_{ijk}^{UAV} \cdot task_i$.

Since residual task amount $task_i$ is dynamically changed in each time slot, e.g., when UAV serves GC i in slot t, value of $task_i$ and $c_{ijk}^{UAV} \cdot task_i$ will be smaller, leading to the fact that UAV will find another user with higher $c_{ijk}^{UAV} \cdot task_i$.

The basic idea of Algorithm 2 is pre-scheduling UAVs' trajectory between time slot k and $k + step - 1$ while satisfying the constraints of UAV's maximum flight speed (C4), UAV maximum communication range (C5), and safety flight distance between two neighboring UAVs (C6) proposed in Section 3.2. Then, the algorithm adjusts UAVs' trajectories calculated in the step above to shorten the total flight distance of UAVs.

Specifically, Algorithm 2 first finds candidate locations of UAVs from GCs' location u that satisfies maximum speed constraint (C4)

Algorithm 2: TDA (U, R, K)

Input: Initial location of GCs u_{ik}
Communication range of UAV R_j
Current time slot k
Residual task amount $task$
Initial location for UAVs q_{i1}
Output: UAVs' trajectory q

1 $Temp = 0$
2 **for** $j \in J$ **do**
3 **for** *st=1 to step* **do**
4 $Cand = \{u_{ik} \mid dis(u_{ik}, q_{jk}) \le v_{max}\}, i \in I$
5 Delete $i \in Cand$ that does not satisfy the safety-distance constraint (C6);
6 **for** $j' \in Cand$ **do**
7 $served = \{u_{ik} \mid dis(u_{ik}, Cand_{j'}) \le R_{j'}\}, i \in I$
8 Calculate the corresponding transmission rate $c^{UAV}_{ij'k}$.
9 **end**
10 $j^* = argmax(c_{ij^*k} \cdot task_i), i \in served, j^* \in Cand$
11 $temp_{j(k+st-1)} = Cand_{j^*}$
12 **end**
13 Adjust $temp_{j(k+st-1)}$ in order to shorten the flight length
14 $q_j = q_j \cup \{temp_{jk}, ..., temp_{j(k-1+st)}\}$;
15 **end**

(line 4) and saves in the candidate set $Cand$. Then, all the locations u_{ik} in set $cand$ are checked, and some that do not obey the safety distance constraint (C6) are deleted from $Cand$. Afterward, Algorithm 2 retrieves all UAVs' candidate locations j' in set $Cand$ and selects one with maximum residual transmission rate to GCs (line 6 to line 10). The selected UAVs' location at the current slot $k+st-1$ is then saved in set $temp$ as the candidate UAV's trajectory (line 11). When all preschedule steps end, UAVs' locations $temp$ are modified to shorten the total flight distance of each UAV by resorting to locations in set $temp$.

Lemma 1. *For each UAV $j \in J$, the time complexity of TDA is $O(m^2)$ step.*

Proof. We analyze the time complexity for each UAV $j \in J$, for TDA runs in parallel for each UAV.

First, algorithm TDA first iterates steps $st \in [1, step]$ to work out the candidate locations for UAV $cand_j$ (line 3 to line 12), time complexity of this loop is $O(step)$. Second, In lines 4 and line 5, TDA retrieves the locations of all GCs to find the initial candidate set $Cand$ and deletes GCs that do not satisfy the constraint (C6), and the time complexity of this step is $O(m)$. Then, from line 6 to line 10, TDA iterates each element in setting Cand to calculate the corresponding transmission rate $c^{UAV}_{ij'k}$ and select GC i with the maximum residual rate (line 10). The time complexity of this step is $O(m)$. Finally, algorithm TDA sorts the set $temp$ and outputs the final trajectory for UAV j. The time complexity for this step is $O(m \cdot log_2\ m)$.

Overall, for each UAV $j \in J$, time complexity of TDA is $O(step \cdot (m+m) + m \cdot lgm)$, which is $O(m^2)$. ■

4.3. Task allocation strategy

After using the trajectory design algorithm proposed in Section 4.2, UAVs' trajectory q_{jk} in slot k can be calculated. We now focus on the (P2) problem. (P2) is the subproblem of TDTAP, aiming at maximize the task completion rate of all GCs, under the *slot allocation constraints* (Eqs. (C1) (C2) (C3)) and the *maximum amount constraints* (Eq. (C7)).

$$(P2): \max_{x,y,z} D_{sum} \tag{11}$$

$$\text{s.t. (C1) (C2) (C3) (C7)} \tag{12}$$

$$x^{UAV}_{ijk}, y^{BS}_{ibk}, z^{loc}_{ik} \ge 0 \tag{13}$$

Offline algorithms cannot solve the problem (P2) directly because UAVs are unable to obtain GC's location and their task amount until they fly close and get connections with them. Thus, an online algorithm must be proposed.

To better illustrate our algorithm, we define $c_{ihk} = c^{UAV}_{ijk} \cup c^{BS}_{ibk} \cup c^{loc}_{ik}$, where $h = \{0, [1, n], n+1\}$ means all devices including GC itself, UAVs, and the BS, specifically:

$$c_{ihk} = \begin{cases} c^{loc}_{ik}, & h = 0, \\ c^{UAV}_{ijk}, & h = [1, n]. \\ c^{BS}_{ibk}, & h = n+1. \end{cases} \tag{14}$$

Similar to c_{ihk}, we define $x_{ihk} = x^{UAV}_{ijk} \cup x^{BS}_{ibk} \cup x^{loc}_{ik}$, which is written in Eq. (15).

$$x_{ihk} = \begin{cases} x^{loc}_{ik}, & h = 0, \\ x^{UAV}_{ijk}, & h = [1, n], \\ x^{BS}_{ibk}, & h = n+1. \end{cases} \tag{15}$$

Referring to related work [22,31], we apply the primal–dual technology to solve the problem (P2), where the dual problem of (P2) can be transferred as (D2) and α, β, γ, and σ are corresponding dual variables to constraint (C1)∼(C3) and (C7) in problem (P2). We propose a competitive online algorithm TAS in *Algorithm 3*.

$$(D2): \min_{\alpha,\beta,\gamma} \sum_{i=1}^{m} c_i \alpha_i + \sum_{j=1}^{n} \sum_{k=1}^{s} \beta_{jk} + \sum_{k=1}^{s} \gamma_k + \sum_{i=1}^{I} \sum_{k=1}^{s} \sigma_{ik} \tag{16}$$

$$\text{s.t. } c^{UAV}_{ijk} \cdot \alpha_i + \beta_{jk} + \sigma_{ik} \ge c^{UAV}_{ijk} \tag{17}$$

$$c^{BS}_{ibk} \cdot \alpha_i + \gamma_k + \sigma_{ik} \ge c^{BS}_{ibk} \tag{18}$$

$$c^{loc}_{ik} \cdot \alpha_i + \sigma_{ik} \ge c^{loc}_{ik} \tag{19}$$

$$\alpha, \beta, \gamma \ge 0 \tag{20}$$

The basic idea of TAS is described as follows: GCs prefer to offload their tasks to nearby UAVs, and UAVs process tasks from the GC with a maximum residual transmission rate. Then, GCs that are not served by UAVs choose to offload their tasks to the BS when the transmission rate between GCs and the BS is larger than the computation rate of the GC itself. Otherwise, GCs process the task locally.

In particular, each UAV j serves the GC i that has a maximum residual transmission rate and delivers the total c^{UAV}_{ijk} data to the GC i (line 5 to line 9). UAVs select the served GCs in the current slot k, and the rest GCs offload tasks to the BS or directly complete them (line 12 to line 23). The algorithm TAS compares the task processing amount in the local computing of the rest GCs with the transmission rate between the BS and GCs. If the transmission rate is larger than the local computing rate, GC i offloads tasks to the BS (line 13 to line 17); otherwise, GC i directly computes the charges (line 18-line 22).

Finally, for each GC $i \in I$, when device j (including UAVs, the BS or GC itself) obtains c_{ijk} task from GC i, α_i will be updated as $\alpha_i = \alpha_i(1 + \frac{\sum_{j \leftrightarrow i} c_{ijk}}{c_i}) + \frac{\sum_{j \leftrightarrow i} c_{ijk}}{(d-1)c_i}$, where $j \leftrightarrow i$ indicates UAV j serves GC i at current slot (line 26). Here d is a value used in the calculation, and we define this value in Lemma 1.

In Algorithm 3, variable $\beta_{jk}, \gamma_k, \sigma_{ik}$ only updates at each time slot k, but variable α_i probably updates in many different time slots. We note that dual variable α representing completing ratio of task i satisfies the condition of $\alpha_i \le 1$.

Lemma 2. *To meet $\alpha_i \le 1$, value of d should be set as $d = (1 + \frac{1}{c_{min}})^{c_{min}}$, where $c_{min} = min\ c_i$.*

Proof. We define variable α^k_i as the value of α_i in slot k, written in detail as:

$$\alpha^k_i = \alpha^{k-1}_i (1 + \frac{\sum_{h \leftrightarrow i} c_{ihk}}{c_i}) + \frac{\sum_{h \leftrightarrow i} c_{ihk}}{(d-1)c_i} \tag{21}$$

Algorithm 3: TAS (U, R, K)

Input: Transmission rate of UAV c_{ijk}^{UAV}
Transmission rate of BS c_{ijk}^{BS}
Computation rate of GC c_{ik}^{loc}
Data amount for each GC c_i
Current slot $k \in K$
Output: variable $x, y, z, \alpha, \beta, \gamma$
Total throughput $data$

1 $c_{min} = min_i\ c_i$
2 $h = (1 + \frac{1}{c_{min}})^{c_{min}}$
3 $x = 0, y = 0, z = 0, \alpha = 0, \beta = 0, \gamma = 0$
4 $s = \emptyset$
5 **for** $j \in J$ **do**
6 $\quad i^* = argmax_{i \in I}(c_{i^*jk}^{UAV}(1 - \alpha_{i^*}))$
7 $\quad$ **if** $(c_{i^*jk}^{UAV}(1 - \alpha_i) > 0$ *and* $c_{i^*jk}^{UAV} > max(c_{i^*bk}^{BS}, c_{ik}^{loc}))$ **then**
8 $\quad\quad x_{i^*jk}^{UAV} = 1$
9 $\quad\quad \beta_{jk} = (c_{i*jk}^{UAV} - c_{ik}^{loc})(1 - \alpha_{i^*})$
10 $\quad\quad data = data + c_{i*jk}^{UAV}$
11 $\quad$ **end**
12 $\quad S = S \cup \{i^*\},\ I' = I/\{i^*\}$
13 **end**
14 **for** *each* $i' \in I'$ **do**
15 $\quad$ **if** $c_{i'bk}^{BS} > c_{ik}^{loc}$ **then**
16 $\quad\quad y_{i'bk} = 1$
17 $\quad\quad \gamma_k = (c_{i^*bk}^{BS} - c_{ik}^{loc})(1 - \alpha_{i^*})$
18 $\quad\quad data = data + c_{i^*bk}^{BS}$
19 $\quad$ **else**
20 $\quad\quad z_{i'k}^{loc} = 1$
21 $\quad\quad \sigma_{i'k} = c_{i'k}^{loc}(1 - \alpha_i)$
22 $\quad\quad data = data + c_{i'k}^{loc}$
23 $\quad$ **end**
24 **end**
25 **for** *each* $i \in I$ **do**
26 $\quad \alpha_i = \alpha_i(1 + \frac{\sum_{h \leftrightarrow i} c_{ihk}}{c_i}) + \frac{\sum_{h \leftrightarrow i} c_{ihk}}{(d-1)c_i}$
27 $\quad$ **if** $\sum_{h \leftrightarrow i} \sum_k c_{ihk} > c_i$ **then**
28 $\quad\quad x_{ihk} = \frac{c_i - \sum_{h \leftrightarrow i} \sum_{k=1}^{K-1} c_{ihk}}{\sum_{j \leftrightarrow i} c_{ihk}}$
29 $\quad$ **end**
30 **end**

We define $c_{min} = min\ c_i$, and in Eq. (21), α_i is set as a geometric progression with initial value of $\frac{\sum_{h \leftrightarrow i} c_{min}}{(d-1)c_i}$ and iterative ratio of $1 + \frac{\sum_{h \leftrightarrow i} c_{min}}{c_i}$. Then, the general term formula of α_i can be written as the sum of geometric progression, which is:

$$\alpha_i^k = \frac{(\frac{\sum_{h \leftrightarrow i} c_{ijk}}{c_{min}})^{\frac{c_{min}}{1}}}{d-1} \leq 1 \tag{22}$$

For each GC $i \in I$ in slot k, $\sum_{j \leftrightarrow i} c_{ijk} = 1$, value h is calculated as:

$$d = (1 + \frac{1}{c_{min}})^{c_{min}} \quad \blacksquare \tag{23}$$

Theorem 1. *Algorithm 3 is* $\frac{e}{e-1}$*-competitive.*

Proof. The solution to the problem (P2) is defined as:

$$P_{sol} = \max D_{sum} \tag{24}$$

Moreover, the solution to the dual problem (D2) is denoted as:

$$D_{sol} = \min_{\alpha,\beta,\gamma} \sum_{i=1}^{m} c_i \alpha_i + \sum_{j=1}^{n} \sum_{k=1}^{s} \beta_{jk} + \sum_{k=1}^{s} \gamma_k + \sum_{i=1}^{I} \sum_{k=1}^{s} \sigma_{ik} \tag{25}$$

The optimal solution of *Algorithm 3* is represented as opt.

Based on weak duality theory in operational research [31], the maximum value of primal solution P_{sol} cannot exceed the optimal solution opt and minimum value of dual solution D_{sol} cannot be lower than optimal solution opt. Thus, the relationship between primal solution P_{sol}, optimal solution opt and dual solution D_{sol} is written in Eq. (26).

$$P_{sol} \leq opt \leq D_{sol} \tag{26}$$

The approximate ratio ω is then written as:

$$\omega = \frac{opt}{P_{sol}} \leq \frac{D_{sol}}{P_{sol}} = \frac{\Delta D}{\Delta P} \tag{27}$$

where ΔD and ΔP are increment of D_{sol} and P_{sol} for GC i between two neighboring time slot $k-1$ and k in Algorithm 3. We use ratio $\frac{\Delta D}{\Delta P}$ to illustrate the upper bound of competitive ratio ω.

When GC i selects device h to process its tasks at slot k, variable x_{ihk} is increased from 0 to 1. Based on Eq. (11), the value of ΔP is:

$$\Delta P = \sum_{h \leftrightarrow i} c_{ihk} x_{ihk} = \sum_{j \leftrightarrow i} c_{ihk} \tag{28}$$

Meanwhile, based on Eq. (16) (object function of (D2)), we analyze the growth of ΔD by each variable. Specifically, for each GC i, the growth of $\sum_{i=1}^{m} c_i \alpha_i$ from time slots $k-1$ to k is:

$$c_i \cdot \Delta\alpha_i = c_i \cdot (\alpha_i^k - \alpha_i^{(k-1)}) \tag{29}$$

the growth of $\sum_{j=1}^{n} \sum_{k=1}^{s} \beta_{jk}$ from time slots $k-1$ to k is:

$$\sum_{j=1}^{n} [\Delta(\sum_{k=1}^{s} \beta_{jk})] = \sum_{j=1}^{n} \beta_{jk} \tag{30}$$

the growth of $\sum_{k=1}^{s} \gamma_k$ from time slots $k-1$ to k is:

$$\Delta(\sum_{k=1}^{s} \gamma_k) = \gamma_k \tag{31}$$

the growth of $\sum_{i=1}^{I} \sum_{k=1}^{s} \sigma_{ik}$ from time slots $k-1$ to k is:

$$\Delta(\sum_{k=1}^{s} \sigma_{ik}) = \sigma_{ik} \tag{32}$$

Based on Eqs. (29) to (32), the value of ΔD is:

$$\begin{aligned}
\Delta D &= \sum_{j=1}^{n} \beta_{jk} + \gamma_k + \sigma_{ik} + c_i \cdot (\alpha_i^k - \alpha_i^{(k-1)}) \\
&= \sum_{j \leftrightarrow i} (c_{ijk}^{UAV} - c_{ik}^{loc}) \cdot (1 - \alpha_i^{k-1}) \\
&\quad + \sum_{b \leftrightarrow i} (c_{ibk}^{BS} - c_{ik}^{loc}) \cdot (1 - \alpha_i^{k-1}) \\
&\quad + c_{ik}^{loc}(1 - \alpha_i^{k-1}) \\
&\quad + c_i \cdot (\alpha_i^{k-1} \cdot \frac{\sum_{j \leftrightarrow i} c_{ijk}}{c_i} + \frac{\sum_{j \leftrightarrow i} c_{ijk}}{(d-1)c_i})
\end{aligned}$$

In each slot $k \in K$, GC i only choose one device (including UAVs, the BS, and GC i itself) to complete its data according to line 5 to 23 in Algorithm 3. To analyze the value of ΔD, we separately calculate ΔD when GC i offloads its task to UAVs, the BS, or computes the task by itself.

Case 1: If GC i offloads its tasks to UAVs, the equation will be transferred as:

$$\begin{aligned}
\Delta D &= c_i \cdot \alpha_i^k + \sum_{j=1}^{n} \beta_{jk} \\
&= \sum_{j \leftrightarrow i} (c_{ijk}^{UAV} - c_{ik}^{loc}) \cdot (1 - \alpha_i^{k-1}) \\
&\quad + c_i \cdot (\alpha_i^{k-1} \cdot \frac{\sum_{j \leftrightarrow i} c_{ijk}}{c_i} + \frac{\sum_{j \leftrightarrow i} c_{ijk}}{(d-1)c_i}) \\
&\leq \sum_{j \leftrightarrow i} c_{ijk}^{UAV} \cdot (1 - \alpha_i^{k-1}) \\
&\quad + c_i \cdot (\alpha_i^{k-1} \cdot \frac{\sum_{h \leftrightarrow i} c_{ihk}}{c_i} + \frac{\sum_{h \leftrightarrow i} c_{ihk}}{(d-1)c_i})
\end{aligned}$$

Since $c_{ihk} = c_{ijk}^{UAV}$ in this case, value of ΔD is written as:

$$\begin{aligned}\Delta D &\le \sum_{h\leftrightarrow i} c_{ihk}\cdot(1-\alpha_i^{k-1}) \\ &\quad + c_i\cdot(\alpha_i^{k-1}\cdot\frac{\sum_{h\leftrightarrow i}c_{ihk}}{c_i}+\frac{\sum_{h\leftrightarrow i}c_{ihk}}{(d-1)c_i}) \\ &\le \sum_{h\leftrightarrow i} c_{ihk}(1+\frac{1}{d-1})\end{aligned}$$

Therefore, the competitive ratio of **Case 1** is:

$$\begin{aligned}\omega &= \frac{Opt}{P_{sol}} \le \frac{D_{sol}}{p_{sol}} = \frac{\Delta D}{\Delta P} \\ &= \frac{(1+\frac{1}{d+1})\sum_{j\leftrightarrow i}c_{ihk}}{\sum_{j\leftrightarrow i}c_{ihk}} \\ &= 1+\frac{1}{d-1}\end{aligned}$$

Case 2: If GC i offloads its tasks to the BS, the equation will be transferred as:

$$\begin{aligned}\Delta D &= c_i\cdot\alpha_i^k+\gamma_k \\ &= \sum_{b\leftrightarrow i}(c_{ibk}^{BS}-c_{ik}^{loc})\cdot(1-\alpha_i^{k-1}) \\ &\quad + c_i\cdot(\alpha_i^{k-1}\cdot\frac{\sum_{h\leftrightarrow i}c_{ihk}}{c_i}+\frac{\sum_{h\leftrightarrow i}c_{ihk}}{(d-1)c_i}) \\ &\le \sum_{b\leftrightarrow i} c_{ibk}^{BS}\cdot(1-\alpha_i^{k-1}) \\ &\quad + c_i\cdot(\alpha_i^{k-1}\cdot\frac{\sum_{h\leftrightarrow i}c_{ihk}}{c_i}+\frac{\sum_{h\leftrightarrow i}c_{ihk}}{(d-1)c_i})\end{aligned}$$

Since $c_{ihk} = c_{ibk}^{BS}$ in this case, similar to Case 1, ΔD is written as

$$\begin{aligned}\Delta D &\le \sum_{h\leftrightarrow i} c_{ihk}\cdot(1-\alpha_i^{k-1}) \\ &\quad + c_i\cdot(\alpha_i^{k-1}\cdot\frac{\sum_{h\leftrightarrow i}c_{ihk}}{c_i}+\frac{\sum_{h\leftrightarrow i}c_{ihk}}{(d-1)c_i}) \\ &\le \sum_{h\leftrightarrow i} c_{ihk}(1+\frac{1}{d-1})\end{aligned}$$

Therefore, the competitive ratio of **Case 2** is

$$\omega = \frac{opt}{P_{sol}} \le \frac{D_{sol}}{p_{sol}} = \frac{\Delta D}{\Delta P} = 1+\frac{1}{d-1}$$

Case 3: If GC i computes the tasks in local, the equation will be transferred as:

$$\begin{aligned}\Delta D &= c_i\cdot\alpha_i^k+\sigma_{ik} \\ &= c_{ik}^{loc}(1-\alpha_i^{k-1}) \\ &\quad + c_i\cdot(\alpha_i^{k-1}\cdot\frac{\sum_{h\leftrightarrow i}c_{ihk}}{c_i}+\frac{\sum_{h\leftrightarrow i}c_{ihk}}{(d-1)c_i}) \\ &\le \sum_{h\leftrightarrow i} c_{ihk}(1+\frac{1}{d-1})\end{aligned}$$

Therefore, the competitive ratio of **Case 3** is

$$\omega = \frac{opt}{P_{sol}} \le \frac{D_{sol}}{p_{sol}} = \frac{\Delta D}{\Delta P} = 1+\frac{1}{d-1}$$

According to Lemma 2, $d = (1+\frac{1}{c_{min}})^{c_{min}}$. Since $lim_{c_{min}\to\infty}(1+\frac{1}{c_{min}})^{c_{min}} = e$, the value of competitive ratio ω is:

$$\begin{aligned}\omega &= 1+\frac{1}{e-1} \\ &= \frac{e}{e-1}\end{aligned}$$

■

Lemma 3. *The algorithm TAS can solve problem P2 in the $O(s^2)$ step.*

Proof. In each iteration, algorithm TAS attempts to search all the time slots $k \in K$ and UAVs $j \in J$ to find the most suitable UAV for GC i. The time complexity of this step is $O(s\cdot n)$. Then, the algorithm offloads GCs' tasks to the BS or calculates them locally, taking time $O(s\cdot m)$. Finally, the algorithm updates the variable α, taking time $O(s\cdot m)$. Overall, the time complexity of algorithm TAS is $O(s\cdot(n+m+m))$, namely $O(s^2)$. ■

5. Performance evaluation

In this section, we evaluate the performance of our algorithm MUTAA by comparing the offline algorithm OPT, the online algorithm Round-Robin, and the algorithm HOTSPOT.

5.1. Simulation settings

Simulation settings are based on Ref. [18,32]. The total simulation time s is set as 20 to 200 slots, where the length of a single slot Δt is set as 100 ms. There are 1 to 20 UAVs in the experiment; each of them flies at a fixed altitude $H = 20$ m with a maximum speed of $v_{max} = 40$ m/s, and keeps the safe distance $dis_{min} = 15$ m from other UAVs. A computational edge node is mounted on a UAV to process tasks offloaded from GCs, and UAVs are only allowed to serve GCs in their communication ranges. GC i contains c_i amount of computation task, whose value ranges from 10 MB to 30 MB, and has a local computation rate c_{ik}^{loc} ranging from 0.05 MB/s to 0.1 MB/s. It can offload tasks to nearby UAVs or the remote BS for computation and communication assistance. We define the maximum network bandwidth as 3 MHz, and the transmit power of GC i is set as $p_i = 0.5$ W. Based on the above definitions, the transmission rate between the BS and each GC i is set as c_{ibk}^{UAV}, whose value can be calculated in advance.

GCs' mobility model is referred to paper [33], where vehicles' velocities are randomly generated using a truncated Gaussian distribution with a mean equal 70 km/h, variance 16 km/h, and velocities can be varied between 50~90 km/h, where the vehicles are driving in this speed with one direction randomly generated at the first time slot. The vehicle will turn around and return if it meets the scenario border.

Based on previous settings, we compare the proposed algorithm MUTAA with the offline optimal algorithm OPT, the online Round-Robin policy, a trajectory design algorithm named HOTSPOT [32], a special case of MUTAA named SINGLE, details described as follows:

- **OPT**: OPT directly solves the offline version of the problem TD-TAP by mathematic tools, in which UAVs know locations of all GCs and transmission rates c_{ibk}^{BS}, c_{ijk}^{UAV} in advance.
- **Round-Robin**: In Round-Robin, UAV j evenly distributes its communication resource to all GCs within the communication range of R_j at each time slot. The competitive ratio of the Round-Robin policy is proved to be 2 [22].
- **HOTSPOT**: HOTSPOT [32] utilizes maximum clique technology to deploy UAVs in proper locations, and decides the GCs' task allocations according to the average offloading probability.
- **SINGLE**: A special case of MUTAA when the preschedule step is set as 1. The algorithm SINGLE is proposed to evaluate the performance of MUTAA via different preschedule steps.

On this basis, simulations are given in the following sections to evaluate the performance of task processing amount and average running time for a single time slot $\Delta t = 100$ ms. For these settings, we generate 10 scenarios for each simulation and take the average number as the final results.

5.2. Impact of GC numbers m

Several experiments discuss the relationship between task processing amount and GC numbers in this part. In this scenario, 3 UAVs have a communication range of $R_j = 50$ m, with a maximum speed of 40 m/s and a safety distance of 5 m. There are 30 to 1000 GCs randomly generated in 300 m × 300 m area, with task amount of c_i varying

Table 2
Task processing amount with various GC numbers m ($s = 100$, $n = 3$).

GC number	Task processing amount (MB)			Competitive ratio		Average running time in Δt (ms)		
	Round-Robin	MUTAA	OPT	Round-Robin	MUTAA	Round-Robin	MUTAA	OPT
30	426.34	**483.86**	521.83	1.22	**1.08**	0.462	**1.103**	47.28
60	786.20	**944.83**	1312.38	1.67	**1.39**	0.913	**1.843**	128.23
90	966.31	**1214.56**	1735.86	1.79	**1.43**	1.663	**3.707**	322.12
120	1314.69	**1416.28**	1987.52	1.51	**1.40**	2.068	**4.675**	1203.41
150	1541.26	**1867.76**	2419.15	1.57	**1.29**	4.318	**6.193**	4703.41
200	1752.78	**2342.34**	2831.65	1.62	**1.20**	8.726	**10.665**	24 163.28
300	2659.82	**3520.39**	–	–	–	13.643	**16.817**	–
500	3391.35	**4765.12**	–	–	–	21.322	**25.134**	–
1000	5611.17	**8868.26**	–	–	–	32.673	**47.374**	–

Table 3
Task processing amount with various UAV numbers n ($s = 100$, $m = 100$).

UAV number	Task processing amount (MB)			Competitive ratio		Average running time in Δt (ms)		
	Round-Robin	MUTAA	OPT	Round-Robin	MUTAA	Round-Robin	MUTAA	OPT
1	654.13	**821.29**	1276.27	1.95	**1.56**	0.612	**0.791**	39.26
2	966.23	**1138.51**	1673.29	1.73	**1.47**	1.391	**1.329**	57.35
3	1124.39	**1451.23**	1878.12	1.67	**1.29**	2.722	**4.069**	180.810
4	1289.32	**1653.31**	2041.75	1.58	**1.23**	3.606	**5.7928**	431.843
5	1396.32	**1795.34**	2141.54	1.53	**1.19**	5.3667	**7.8041**	1203.41
8	1479.91	**1923.36**	2217.95	1.49	**1.15**	8.095	**11.761**	6163.28
10	1657.67	**2020.70**	–	–	–	12.544	**16.9073**	–
15	1734.23	**2176.98**	–	–	–	21.256	**32.918**	–
20	1761.27	**2208.87**	–	–	–	30.954	**56.074**	–

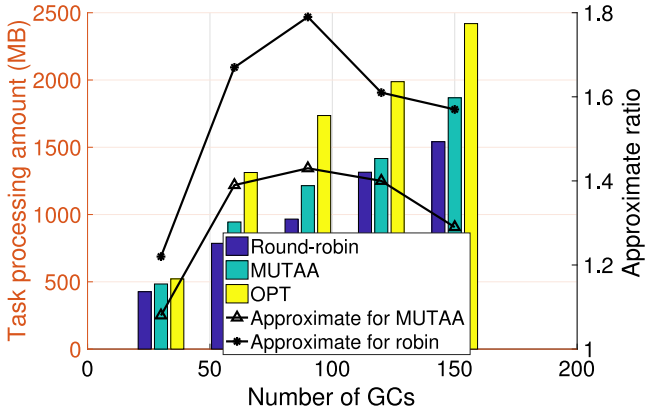


Fig. 2. Task processing amount versus GCs' number.

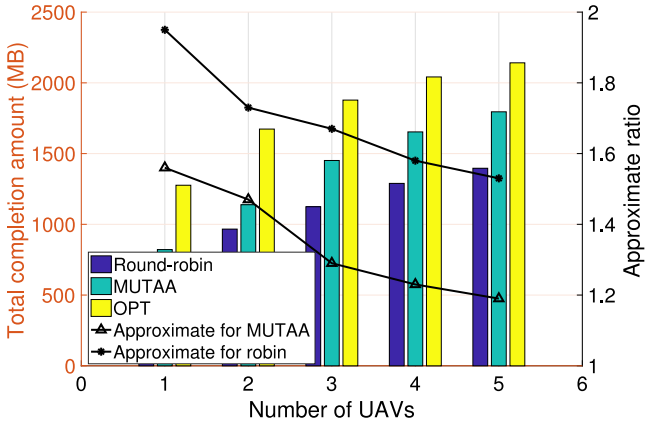


Fig. 3. Task processing amount versus UAVs' number.

from 15 MB to 30 MB. The BS is set in the center of the scenario, and the preschedule step for MUTAA is set as 5. We experiment to evaluate the task processing amount of MUTAA, OPT, and Round-Robin, respectively. Experimental results are shown in Fig. 2 and Table 2.

It can be seen from Fig. 2 that the three algorithms Round-Robin, MUTAA, and OPT complete more tasks when the number of GCs increases because the more GCs are, the more tasks are produced. MUTAA completes more tasks than Round-Robin, which does not take transmission rate c_{ijk}^{UAV} and residual task amount into consideration. OPT performs better than MUTAA since OPT knows the transmission rate and locations of all GCs, UAVs, and the BS in advance. Results also show that when the GCs number reaches 200, MUTAA completes 83% tasks on average of OPT, and the competitive ratio between MUTAA and OPT is less than 1.58, while the competitive ratio between Round-Robin and OPT is less than 2.

However, OPT spends much more time achieving the final results. In Table 2, it can be seen that when the number of GCs grows, the execution time of OPT increases exponentially. When the number of GCs reaches 200, OPT spends 42.16 s on average to achieve the results at each time slot ($\Delta t = 100$ ms), which is too long and cannot be applied online. By contrast, the running time of MUTAA only costs 10.67 ms at each time slot.

5.3. Impact of UAV numbers n

This section provides a group of experiments to show the relationship between task processing amount and UAV numbers. Specifically, 100 GCs generated in 300×300 m areas, with task amount of c_i varying from 15 MB to 30 MB. The BS is set in the center of the scenario, and the preschedule step for the algorithm MUTAA is set as 5. UAV numbers vary from 1 to 20, each of which has a communication range $R_j = 50$ m, with a maximum speed of 40 m/s and a safety distance of 5 m. We experiment to evaluate the total task amount of the algorithm MUTAA, OPT, and Round-Robin. Experimental results are shown in Fig. 3 and Table 3.

It can be seen from Fig. 3 and Table 3 that the three algorithms Round-Robin, MUTAA, and OPT, complete more tasks when the number of UAVs increases. However, with the number of UAVs growing, the increment of task competition amount decreases. For instance, when UAV numbers increase from 15 to 20, the task processing amount increases to only 31.2MB. That is because the number of GCs in this experiment is fixed, and the total amount of tasks does not change much. Therefore, the competitive ratios of both Round-Robin and

Table 4
Task processing amount with various total simulation time slots s ($c_i = 9$MB, $m = 100$).

Slot number	Task processing amount (MB)			Competitive ratio		Average running time in Δt (ms)		
	Round-Robin	MUTAA	OPT	Round-Robin	MUTAA	Round-Robin	MUTAA	OPT
40	532.51	**632.20**	837.31	1.57	**1.34**	0.3870	**0.5414**	38.314
80	642.13	**717.18**	900.00	1.40	**1.25**	0.2743	**0.5925**	168.241
120	708.75	**762.56**	900.00	1.27	**1.18**	0.1376	**0.6695**	283.41
160	724.29	**790.46**	900.00	1.24	**1.13**	0.1379	**0.9038**	1975.56
200	745.69	**810.92**	900.00	1.21	**1.11**	0.1488	**1.5894**	3534.17

Table 5
Task processing amount with various total simulation time slots K ($c_i = 20$MB, $m = 100$).

Slot number	Task processing amount (MB)			Competitive ratio		Average running time in Δt (ms)		
	Round-Robin	MUTAA	OPT	Round-Robin	MUTAA	Round-Robin	MUTAA	OPT
40	781.93	**880.53**	1259.74	1.61	**1.43**	0.4638	**0.5417**	4.9727
80	930.74	**1033.57**	1492.35	1.64	**1.44**	0.1462	**0.9391**	12.7359
120	1097.86	**1256.46**	1857.34	1.69	**1.48**	0.5522	**1.8698**	56.0810
160	1320.09	**1408.35**	2000.00	1.52	**1.42**	0.7306	**2.7928**	103.1843
200	1352.27	**1535.41**	2000.00	1.47	**1.30**	1.3667	**4.8041**	510.3417

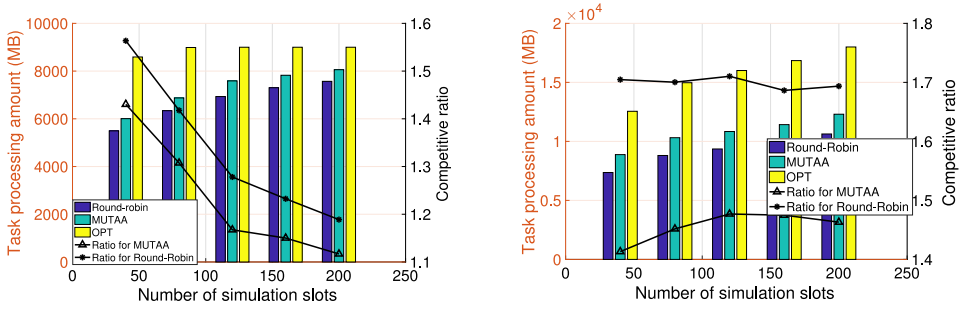


Fig. 4. Task processing amount versus simulation time.

MUTAA decrease when UAV numbers increase because more UAVs are dispatched to assist GCs in completing their tasks.

Simulation results also show that the running time of three algorithms increases when the UAV number grows, for all these algorithms iterates each UAV $j \in J$ to calculate the task processing amount. When comparing the three algorithms, OPT's running time is much longer than that of MUTAA and Round-Robin, reaching 6.613 s, which is much larger than the given time slot of 100 ms and cannot be applied in online development. On the contrary, the running time of MUTAA only costs 11.76 ms at each time slot.

5.4. Impact of total simulation slots s

In this section, we provide a group of experiments to illustrate the relationship between task processing amount and total simulation time slots s. We conduct two experiments to illustrate situations where each GC's task amount differs. Specifically, there are 100 GCs randomly generated in the 300×300 m area, with the task amount of $c_i = 9$ MB in the first experiment and $c_i = 20$ MB in the second experiment. The BS is set in the center of the area, and the preschedule step for the algorithm MUTAA is set as 5. There are 5 UAVs in this scenario, and each has a communication range of $R_j = 50$ m, with a maximum speed of 40 m/s and a safety distance of 5 m. We conduct the two experiments to change the simulation time from 10 to 140 slots and evaluate the task processing amount of the algorithm MUTAA, the algorithm OPT, and the algorithm Round-Robin respectively. Experimental results are shown in Table 4, Fig. 4(a), Table 5, and Fig. 4(b).

Table 4 shows how the algorithm MUTAA works when the task amount of each GC is 9 MB. When the number of time slots varies from 40 to 200, the competitive ratio between MUTAA and OPT is less than 1.58, verifying that the algorithm MUTAA is 1.58-competitive. When we compare different simulation times, it can be seen that MUTAA performs better when the simulation time is longer, for OPT completes all tasks when the slot number is 80, and the competitive ratio between MUTAA and OPT is better when the number of slots increases.

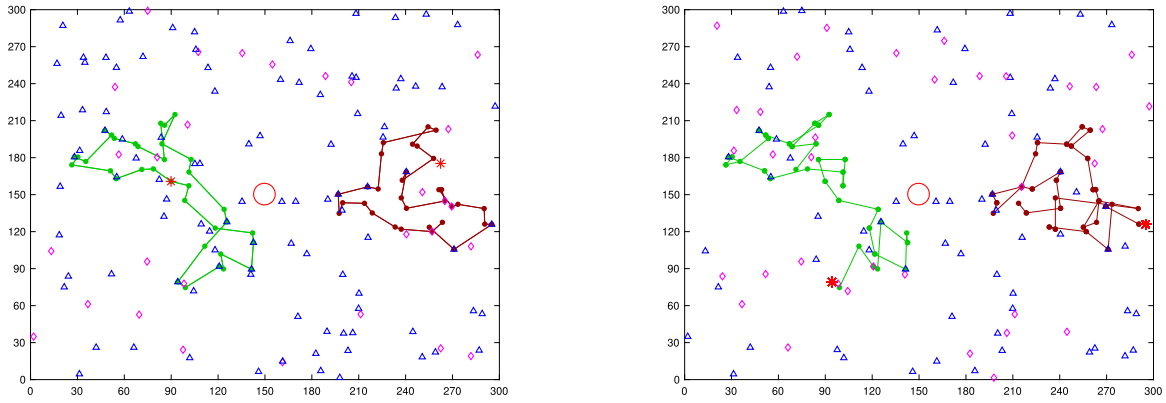
Table 5 shows how MUTAA works when the task amount of each GC is 20 MB. When the number of time slots varies from 40 to 200, the competitive ratio between MUTAA and OPT is less than 1.58, demonstrating that the algorithm MUTAA is 1.58-competitive. When we compare different time slots, it can be seen that the competitive ratio of MUTAA first increases when the slot number is smaller than 120 because the accumulation gap between MUTAA and OPT is larger when the number of time slots increases. When the slot number is larger than 120, the competitive ratio of MUTAA decreases. That is because OPT already completes all tasks generated by GCs, while MUTAA still has more tasks to complete.

5.5. Impact of different trajectories

The three sections above evaluate task-allocation algorithms between MUTAA and the compared algorithms. In this section, we compare the three trajectory-design algorithms between MUTAA and HOTSPOT [32] and SINGLE.

Specifically, 100 GCs randomly generated in the 300×300 m area, with the task amount c_i varying from 15 MB to 30 MB. There are 3 UAVs in the scenario, each with a communication range $R_j = 50$ m, a maximum speed of 40 m/s, and a safety distance of 5 m. The step for the algorithm MUTAA is set as 5, and that for SINGLE is set as 1. We conduct an experiment to evaluate task process amount of MUTAA and HOTSPOT in 150 time slots. Experimental results are shown in Table 6.

We first compare the task processing amount between MUTAA and HOTSPOT. It can be seen from Table 6 that the task processing amount



(a) MUTAA (b) SINGLE

Fig. 5. UAV's trajectory among MUTAA and SINGLE. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Experimental results for different trajectory algorithms with various GCs' numbers ($s = 100$, $n = 3$).

GCs' number	Task processing amount (MB)			Total flight distance (m)		
	HOTSPOT	MUTAA	SINGLE	HOTSPOT	MUTAA	SINGLE
20	398.29	**500.51**	413.82	551.21	**704.71**	751.62
40	728.01	**920.39**	870.68	761.47	**990.00**	976.37
60	960.59	**1115.84**	1022.67	826.97	**1079.98**	1127.77
90	1071.80	**1391.21**	1236.35	1086.51	**1246.85**	1390.46
100	1212.59	**1579.46**	1404.32	1200.37	**1460.47**	1546.77

of MUTAA and SINGLE is larger than that of HOTSPOT, for the task-allocation algorithm in HOTSPOT only considers the average offloading probability of GCs, and does not take the residual task amount for each GC into consideration. Task processing amount of SINGLE is smaller than that of MUTAA, and the gap between the two algorithms is smaller than 10%, for these two algorithms are similar that they both use preschedule strategy to deploy UAVs in proper locations. However, the total flight distance of SINGLE is much larger than that of MUTAA because UAVs use the trajectory algorithm SINGLE sometimes roaming around two specific GCs, which prolongs the total flight distance of UAVs.

Fig. 5(a) and (b) illustrate the trajectories of MUTAA, SINGLE, and the task scheduling for each GC in scenario with 100 GCs, 2 UAVs and a BS. The green and brown lines in Fig. 5 are trajectories for two different UAVs. The red circle represents the BS, which located at the scenario's center. The red star denotes the GCs that UAVs serve, the blue triangle means the GCs that are served by the BS, where the pink diamond is the GC that computes the task locally. It can be seen from Fig. 5(a) that UAV attempts to select GCs with maximum transmission rate and most residual tasks, which is always the GC near each UAV. Besides, Most GCs near the BS choose to upload their tasks to the BS, for the transmission rate between GCs and the BS is large. When GCs are far from the BS, uploading tasks to BS may take much time, and some GCs choose to compute tasks by themselves.

Then, we compare the total flight distance between HOTSPOT with MUTAA. In Table 6, the total flight distance of the proposed algorithm MUTAA is smaller than that of HOTSPOT since MUTAA utilizes a preschedule strategy to find the most suitable deployment positions for UAVs, according to candidate locations of GCs in the current time slot and locations of UAVs in previous time slots. On the contrary, HOTSPOT utilizes the maximum clique technology to classify GCs into several clusters and deploys UAVs in the center of each cluster. When GCs' locations changes, the center of each cluster (UAV's location) does not change much, leading to the result that UAVs only receive tasks from GCs within a limited range. Therefore, the task processing amount of HOTSPOT is much smaller than that of MUTAA.

6. Conclusion

In this paper, we establish a UAV-aided edge computing system to assist GCs in completing their computing tasks. Distinct from most of the existing works, this paper considers an online scenario where UAVs do not obtain information in advance from GCs, e.g., locations and communication quality. In particular, we formulate a trajectory design and task allocation problem named TDTAP in the mathematic model to maximize the task processing amount of all GCs, under trajectory constraints of UAVs and task amount constraints of GCs. On this basis, an online algorithm MUTAA is proposed to solve the problem TDTAP and is decomposed into two sub-algorithms: trajectory design algorithm TDA and task scheduling algorithm TAS. On the one hand, a trajectory design algorithm named TDA utilizes a preschedule strategy to optimize each UAV's trajectory. On the other hand, a competitive online algorithm named TAS is proposed to schedule tasks for each GC. Theoretical analysis proves that TAS is e/(e–1)-competitive. The experimental results demonstrate that MUTAA completes an average 83% task amount of the optimal offline algorithm OPT but costs much less time.

In the future, a more complex UAV-aided edge computing system will be built to consider cooperation strategies among UAVs. We aim to maximize the task processing amount of all GSs by jointly considering UAVs' trajectory and task allocation strategies under the trajectory design constraints, tasks' deadline constraints, and energy consumption constraints.

CRediT authorship contribution statement

Weidu Ye: Investigation, Theoretical analysis, Experiment evaluation, Writing. **Junzhou Luo:** Funding acquisition, Resources, Supervision, Writing – review & editing. **Wenjia Wu:** Investigation, Experiment evaluation, Writing, Validation. **Feng Shan:** Theoretical analysis, Writing, Validation. **Ming Yang:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix A. Transformation procedure between the primal problem and the dual problem

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.comnet.2022.109405.

References

[1] X. Hu, K.-K. Wong, K. Yang, Z. Zheng, UAV-assisted relaying and edge computing: scheduling and trajectory optimization, IEEE Trans. Wireless Commun. 18 (10) (2019) 4738–4752.

[2] Y. Liu, K. Xiong, Q. Ni, P. Fan, K.B. Letaief, UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization, IEEE Internet Things J. 7 (4) (2020) 2777–2790.

[3] Y. Zeng, Q. Wu, R. Zhang, Accessing from the sky: A tutorial on UAV communications for 5G and beyond, Proc. IEEE 107 (12) (2019) 2327–2375.

[4] Y. Zeng, J. Xu, R. Zhang, Energy minimization for wireless communication with rotary-wing UAV, IEEE Trans. Wireless Commun. 18 (4) (2019) 2329–2345.

[5] Z. Yong, Z. Rui, Energy-efficient UAV communication with trajectory optimization, IEEE Trans. Wireless Commun. 16 (6) (2017) 3747–3760.

[6] C. Zhan, Y. Zeng, Aerial-ground cost tradeoff for multi-UAV-enabled data collection in wireless sensor networks, IEEE Trans. Commun. 68 (3) (2020) 1937–1950.

[7] H. Guo, J. Liu, UAV-enhanced intelligent offloading for internet of things at the edge, IEEE Trans. Ind. Inf. 16 (4) (2020) 2737–2746.

[8] J. Zhang, L. Zhou, Q. Tang, E.C. Ngai, X. Hu, H. Zhao, J. Wei, Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing, IEEE Internet Things J. 6 (2) (2019) 3688–3699.

[9] M. Mozaffari, W. Saad, M. Bennis, M. Debbah, Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications, IEEE Trans. Wireless Commun. 16 (11) (2017) 7574–7589.

[10] Y. Zeng, J. Xu, R. Zhang, Energy minimization for wireless communication with rotary-wing UAV, IEEE Trans. Wireless Commun. 18 (4) (2019) 2329–2345.

[11] Y. Wang, Z. Hu, X. Wen, Z. Lu, J. Miao, Minimizing data collection time with collaborative UAVs in wireless sensor networks, IEEE Access 8 (2020) 98659–98669.

[12] J. Gong, T. Chang, C. Shen, X. Chen, Flight time minimization of UAV for data collection over wireless sensor networks, IEEE J. Sel. Areas Commun. 36 (9) (2018) 1942–1954.

[13] Z. Yu, Y. Gong, S. Gong, Y. Guo, Joint task offloading and resource allocation in UAV-enabled mobile edge computing, IEEE Internet Things J. 7 (4) (2020) 3147–3159.

[14] S. Eom, H. Lee, J. Park, I. Lee, UAV-aided wireless communication design with propulsion energy constraint, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[15] Y. Liang, W. Xu, W. Liang, J. Peng, X. Jia, Y. Zhou, L. Duan, Nonredundant information collection in rescue applications via an energy-constrained UAV, IEEE Internet Things J. 6 (2) (2019) 2945–2958.

[16] M. Alzenad, A. El-Keyi, F. Lagum, H. Yanikomeroglu, 3D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage, IEEE Wirel. Commun. Lett. 6 (4) (2017) 434–437.

[17] D. Franklin, NVIDIA jetson TX2 delivers twice the intelligence to the edge, 2017, [Online]. Available: https://developer.nvidia.com/blog/jetson-tx2-delivers-twice-intelligenceedge/.

[18] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, G.Y. Li, Joint offloading and trajectory design for UAV-enabled mobile edge computing systems, IEEE Internet Things J. PP (99) 1.

[19] F. Zhou, Y. Wu, R.Q. Hu, Y. Qian, Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems, IEEE J. Sel. Areas Commun. 36 (9) (2018) 1927–1941.

[20] X. Cao, J. Xu, R. Zhang, Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization, in: 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2018, pp. 1–5.

[21] S. Jeong, O. Simeone, J. Kang, Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning, IEEE Trans. Veh. Technol. 67 (3) (2018) 2049–2063.

[22] H. Deng, I. Hou, On the capacity-performance trade-off of online policy in delayed mobile offloading, IEEE Trans. Wireless Commun. 16 (1) (2017) 526–537.

[23] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, B. Li, Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2287–2295.

[24] H. Tan, Z. Han, X.-Y. Li, F.C. Lau, Online job dispatching and scheduling in edge-clouds, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9.

[25] X. Song, G. Fan, Selecting salient frames for spatiotemporal video modeling and segmentation, IEEE Trans. Image Process. 16 (12) (2007) 3035–3046.

[26] W. Liu, G. Lin, T. Zhang, Z. Liu, Guided co-segmentation network for fast video object segmentation, IEEE Trans. Circuits Syst. Video Technol. 31 (4) (2021) 1607–1617.

[27] G. Quellec, M. Lamard, B. Cochener, G. Cazuguel, Real-time segmentation and recognition of surgical tasks in cataract surgery videos, IEEE Trans. Med. Imaging 33 (12) (2014) 2352–2360.

[28] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, X. Shen, Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization, IEEE Trans. Veh. Technol. 69 (3) (2020) 3424–3438.

[29] Y. Liu, K. Xiong, Q. Ni, P. Fan, K.B. Letaief, UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization, IEEE Internet Things J. 7 (4) (2020) 2777–2790.

[30] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, Joint computation and communication design for UAV-assisted mobile edge computing in IoT, IEEE Trans. Ind. Inf. 16 (8) (2020) 5505–5516.

[31] N. Buchbinder, J.S. Naor, The design of competitive online algorithms via a primal-dual approach, Found. Trends Theor. Comput. Sci. 3 (2) (2009) 93–263.

[32] Z. Liao, Y. Ma, J. Huang, J. Wang, J. Wang, HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile edge computing in 3D space, IEEE Internet Things J. (2021).

[33] M. Samir, D. Ebrahimi, C. Assi, S. Sharafeddine, A. Ghrayeb, Leveraging UAVs for coverage in cell-free vehicular networks: A deep reinforcement learning approach, IEEE Trans. Mob. Comput. 20 (9) (2021) 2835–2847.

Weidu Ye received the Bachelor's degree in Computer Science from Nanjing Forestry University in 2015. He is currently working towards the Ph.D. degree in the School of Computer Science and Engineering in Southeast University, Nanjing, China. His research interests include edge computing and UAV communications.

Junzhou Luo received the BS degree in applied mathematics and the MS and Ph.D. degrees in computer network, all from Southeast University, China, in 1982, 1992, and 2000, respectively. He is a full professor in the School of Computer Science and Engineering, Southeast University. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and he is a member of the ACM and chair of ACM SIGCOMM China. His research interests are next generation network architecture, network security, cloud computing, and wireless LAN.

Wenjia Wu received the B.S. and Ph.D. degrees in computer science in 2006 and 2013, respectively, from Southeast University. He is an associate professor at the School of Computer Science and Engineering in Southeast University. His research interests include wireless and mobile networks.

Feng Shan received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2015. He is currently an associate professor with the School of Computer Science and Engineering, Southeast University. He was a Visiting Scholar with the School of Computing and Engineering, University of Missouri–Kansas City, Kansas City, MO, USA, from 2010 to 2012. His current research interests include energy harvesting, wireless power transfer, swarm intelligence, and algorithm design and analysis.

Ming Yang was born in 1979. He received a Ph.D. degree in computer science from Southeast University, Nanjing, in 2007. Currently, he is a full professor at the School of Computer Science and Engineering in Southeast University, Nanjing, China. His research interests include network security and privacy. He is a member of CCF and ACM, as well as deputy director of Key Laboratory of Computer Network and Information Integration, Ministry of Education.