Hindawi
Mobile Information Systems
Volume 2020, Article ID 8791030, 11 pages
https://doi.org/10.1155/2020/8791030

Research Article

Efficient Multitask Scheduling for Completion Time Minimization in UAV-Assisted Mobile Edge Computing

Bingxin Zhang,[1,2] **Guopeng Zhang**,[2] **Shuai Ma**,[1] **Kun Yang**,[3] **and Kezhi Wang**[4]

[1]*School of Information & Control Engineering, China University of Mining and Technology, Xuzhou, China*
[2]*School of Computer Science & Technology, China University of Mining and Technology, Xuzhou, China*
[3]*School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China*
[4]*Department of Computer & Information Sciences, Northumbria University, Newcastle, UK*

Correspondence should be addressed to Guopeng Zhang; gpzhang@cumt.edu.cn

Received 10 December 2019; Revised 23 January 2020; Accepted 25 February 2020; Published 17 June 2020

Academic Editor: Alessandro Bazzi

Mobile edge computing (MEC) can alleviate the computing resource shortage problem of mobile user equipment (UEs). However, due to long communication distance or the obstruction of big obstacles, the direct communication link may not exist between a UE and a MEC node. It thus hinders the task offloading in MEC. Unmanned aerial vehicles (UAVs) have high degree of mobility and can carry lightweight computation and storage modules. This paper presents a UAV-assisted MEC method, in which the UAV can relay the task-input data of a UE to the MEC node and can also utilize the airborne computation and storage resource to shorten the execution time of the offloaded tasks. Considering the strict order dependency among multiple offloaded tasks, this paper optimizes the task scheduling and the UAV flight path in a joint manner. A heuristic algorithm based on particle swarm optimization (PSO) is also developed to find the optimal solution. The simulation results show that the proposed multitask scheduling method can always find the best tradeoff between the UAV's position and the wireless channel condition. In comparison to the other three baseline scheduling methods, the proposed method can use the minimum execution time to complete all the offloaded tasks.

1. Introduction

With the development of natural language processing [1], computer vision [2], Internet of Vehicles (IoV) [3, 4], and other artificial intelligence (AI) technologies, today's mobile applications become more latency sensitive and more computation intensive. It is a big challenge to execute these mobile applications by user equipment (UEs) with limited computing resource [5]. Newly emerged mobile edge computing (MEC) technology aims to bring computing resource close to UEs, which is considered as an effective way to alleviate the shortage of computing resource [6–9]. However, in some special environments, such as remote farms, emergency relief, and open pit quarry, the UEs are far away from the available MEC facility. Hence, it is difficult for the UEs to benefit from the MEC directly [10].

In recent years, unmanned aerial vehicles (UAVs) have been widely used in wireless networks due to the attractive properties of high mobility, flexible deployment, and low operating cost [11]. The UAVs carrying wireless transceivers and data caches can act as mobile base stations (BSs) or relay nodes to collect the data generated by the ground UEs. In addition, the UAVs carrying lightweight computation servers can also act as MEC facilities to execute the computation-intensive applications for the ground UEs.

In this paper, we consider a new UAV-assisted MEC model, where a UAV is designed as a MEC site as well as a mobile relay. The UAV can execute the applications offloaded by a UE or further offload the applications to a remote BS.

As far as the related works on the UAV-assisted MEC model, the authors in [12] optimized the number of the offloading computation bits, the computation frequencies of users, and the trajectory of the UAV jointly, in order to minimize the energy consumption of UAV. The authors in [13] designed a multi-workflow model in the UAV-assisted MEC system, which allows different user devices to operate in parallel, thereby effectively reducing the hovering time of the UAV. In [14], the authors jointly optimized the position, the time slice, and the task partition of a UAV, with the aim of minimizing the energy consumption of the ground UEs. In [15], the authors investigate the effect of the distance between the UAV and each sensor node (SN) in a wireless power transfer system on the attenuation amounts of the radio frequency signal and the charging efficiency. This paper aims to maximize the sum harvested energy of all SNs and maximize the minimum received energy among all SNs by jointly optimizing the UAV trajectory and SN's scheduling scheme. In [16], a cellular-connected UAV is designed to act as a mobile relay which can help offload partial computation task of the UE to the BS for execution. In order to minimize the weighted sum energy consumption of the UAV and the UEs, the author jointly optimizes the computation and bandwidth resource and the UAV's trajectory.

Although many important issues related to the UAV-assisted MEC have been addressed, there are no concerns about how a UAV and a BS execute a large application in parallel. It is noted that a large application can be divided into a group of computation tasks. These tasks could be executed in parallel but must follow a strict order [17]. Since this parallel computing method can accelerate the execution, the aim of this paper is set as minimizing the length of time to complete a set of tasks (affiliated to application). The main contribution of this work is summarized as follows.

(i) A UAV-assisted MEC model is set up which supports the UAV and the BS to execute a set of computation tasks in parallel. In the proposed model, the strict order dependency among the computation tasks is given priority consideration.

(ii) In order to minimize the completion time of the offloaded computation tasks, we propose a mathematical model to jointly optimize the task scheduling and the UAV flight trajectory.

(iii) A heuristic algorithm based on particle swarm optimization (PSO) is developed to find the optimal task scheduling as well as the optimal trajectory for the UAV.

The rest of this paper is organized as follows. In Section 2, the model of the studied UAV-assisted MEC system is introduced. The task completion time minimization problem is formulated in Section 3. In Section 4, an effective heuristic algorithm is proposed to solve the problem. We show the simulation results in Section 5. Finally, the paper is summarized in Section 6.

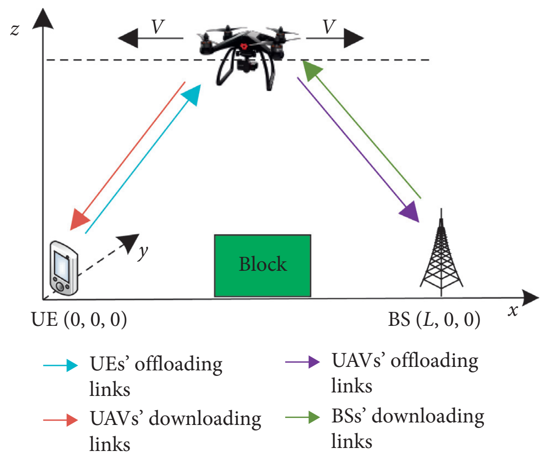

Figure 1: The UAV-assisted MEC system model.

2. System Model

We consider a UAV-assisted MEC system as shown in Figure 1, which consists of a BS, a cellular-connected UAV, and a ground user equipment (UE). The BS has embedded a powerful computing equipment and thus can act as a communication infrastructure as well as a MEC facility. The UE has several computing-intensive applications but cannot process them locally due to the limited computing power. MEC enables the UE to offload the applications to the BS by sending the task-input data to the BS. The BS can execute the applications on behalf of the UE and can also feedback the outcome to the UE.

However, the direct link between the UE and the BS may be very weak due to a long distance or the blockage of big obstacles. In such a situation, the UAV with onboard communication circuit can act as a mobile relay [18] to assist the communication between the UE and the BS. More importantly, the UAV with high mobility can also carry high-performance processor as well as high-capacity data cache. Thus, the UAV could cooperate with the BS to accelerate the completion of the UE's applications.

2.1. Task Partition Model. In general, a computing-intensive application can be portioned into a set of computation tasks in different granularities [19]. These tasks can be processed in parallel, but they must follow a strict execution order. As shown in Figure 2, the precedence relationship of the tasks can be characterized by a directed acyclic task graph $G(V, E)$, where V is the node set and E is the edge set. A node in V represents a computation task. For an application that has been prioritized, a solid line connection with an arrow is used to indicate the priority relationship between the two tasks. That is to say, the task pointed out by the arrow cannot be executed until all its associated previous tasks have been completed. For example, the task 7 in Figure 2 can only be executed after the task 2 and the task 4 have been executed. In Figure 2, the unconnected tasks are independent of each other, such as the tasks 1, 3, and 5. To keep only one start node and one end node in the task

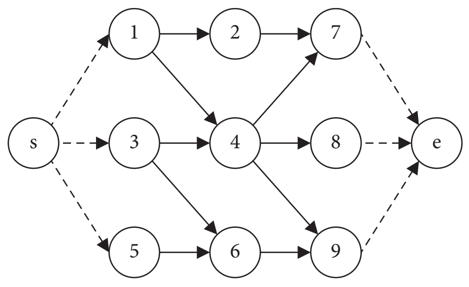

Figure 2: The partition of a large application.

graph, a virtual entry node S and a virtual exit node E are inserted with dashed lines directed to the actual task nodes.

Additionally, in order to support the parallel execution of the multiple tasks, the following assumptions are made as in [19].

(1) All the N tasks in graph G can be portioned into K task-groups. Let $\mathcal{K} = \{1, 2, \ldots, K\}$ denote the set of the task-groups. Within the kth group, the n_k tasks (e.g., nodes 2, 4, and 6 in Figure 2) are independent with each other; i.e., there are no edges between any two of the n_k nodes in graph G. Therefore, the n_k tasks in the k th group can be executed in parallel.

(2) The immediate predecessor of the tasks in the kth group should be completely contained in the $(k-1)$ th group (e.g., nodes 1, 3, and 5 in Figure 2). Thus, the K task-groups in G can be scheduled and performed sequentially.

For the i th task in the kth group, we characterize it by a two-tuple parameter $(b_{k,i}, c_{k,i})$, where $b_{k,i}$ (in bits) denotes the amount of the input data required to execute the task and $c_{k,i}$ (in CPU cycles/bit) denotes the number of CPU cycles required for computing 1 bit of the data. Note that how to partition the application into priority tasks is not our focus; we pay attention to the priority tasks uploading and scheduling in this paper.

2.2. Communication Model. As shown in Figure 1, we project the considered UAV-assisted MEC into a three-dimensional (3D) Euclidean coordinate system. The UE and the BS are located at the coordinates (0, 0, 0) and $(L, 0, 0)$, respectively. The rotary-wing UAV can fly or hover at a fixed altitude $h > 0$ (which could be the minimum altitude required for terrain or building avoidance) and moves with a constant speed $v > 0$ (which satisfies the maximum speed constraint of the UAV). For easy analysis, we assume that the UAV can execute computation tasks while flying but must hover at a fixed position to receive the task-input data (from the UE) or relay the data (to the BS).

The wireless channels from the UAV to the UE and the BS are dominated by Line of Sight (LoS) links, and the Doppler frequency shift due to the UAV's mobility can be perfectly compensated [20, 21]. Let $u_{k,i}^{\mathrm{I}} = (x_{k,i}^{\mathrm{I}}, y_{k,i}^{\mathrm{I}}, h)$ denote the initial position of the UAV just before the kth task-group is scheduled to be executed. Let $u_{k,i}^{\mathrm{C}} = (x_{k,i}^{\mathrm{C}}, y_{k,i}^{\mathrm{C}}, h)$ denote the hovering position for the UAV to receive the task-input data from the UE. Let $u_{k,i}^{\mathrm{R}} = (x_{k,i}^{\mathrm{R}}, y_{k,i}^{\mathrm{R}}, h)$ denote the hovering position for the UAV to forward the task-input data to the BS. It is noted that the initial position of the UAV when it is scheduled to process the tasks in the $(k+1)$th group is just the position where it completes the last task in the kth group. The channel powers from the UAV to the UE and the BS can then be, respectively, given as

$$g_k^{\mathrm{C}} = \frac{\beta_0}{\left(x_k^{\mathrm{C}}\right)^2 + \left(y_k^{\mathrm{C}}\right)^2 + h^2},$$

$$g_k^{\mathrm{R}} = \frac{\beta_0}{\left(x_k^{\mathrm{R}}\right)^2 + \left(y_k^{\mathrm{R}}\right)^2 + h^2}, \tag{1}$$

$$k \in \mathcal{K},$$

where β_0 is the channel power gain at a reference distance $d_0 = 1\,\mathrm{m}$.

Let p^{C} and p^{R} denote the transmit powers at the UE and at the UAV, respectively. Assume that each data link in the system is allocated equal bandwidth B for communication. Then, the maximal data rates for the data uploading link (from the UE to the UAV) and for the relaying link (from the UAV to the BS) can be given as

$$r_k^{\mathrm{C}} = B\log_2\left(1 + \frac{p^{\mathrm{C}} g_k^{\mathrm{C}}}{\sigma^2}\right),$$

$$r_k^{\mathrm{R}} = B\log_2\left(1 + \frac{p^{\mathrm{R}} g_k^{\mathrm{R}}}{\sigma^2}\right), \tag{2}$$

$$k \in \mathcal{K},$$

where σ^2 is the noise power at the receivers of the UAV and the BS, respectively.

3. UAV-Assisted MEC

Due to the limited computing resource, the UE cannot process the computing tasks locally. Hence, it relies on the cooperation between the UAV and the BS to help accomplish the tasks. The computing power of the BS is more powerful than that of the UAV; however, the direct link between the UE and the BS is negligible due to the severe blockage. Hence the BS can only acquire the task-input data through the relaying of the UAV. As mobile applications are generally delay sensitive, it is necessary to account for the time consumption when a task is executed at different positions.

3.1. Executing a Task at the UAV. Once the tasks in the kth task-group is scheduled to be processed, the UE should upload the input data to the UAV first. The time required by the UAV to move from the initial position u_k^{I} to the data collection position u_k^{C} is given by

$$t_{k,i}^{\mathrm{FC}} = \frac{\left\|u_{k,i}^{\mathrm{I}} - u_{k,i}^{\mathrm{C}}\right\|}{v}, \quad k \in \mathcal{K}, \tag{3}$$

The time required by the UAV to collect the input data of the ith task in the kth group can be given by

$$t_{k,i}^{\mathrm{C}} = \frac{b_{k,i}}{r_{k,i}^{\mathrm{C}}}, \quad k \in \mathcal{K}. \tag{4}$$

If this task is scheduled to be processed at the UAV, the execution time can be given by

$$t_{k,i}^{\mathrm{E}} = \frac{\left(b_{k,i}, c_{k,i}\right)}{f_{\mathrm{UAV}}}, \quad k \in \mathcal{K}, \tag{5}$$

where f_{UAV} is the computation capacity of the UAV that is measured by the number of CPU cycles per second.

3.2. Executing a Task at the BS. Otherwise, if the task is scheduled to be processed at the BS, the UAV should cache the task-input data and then find and move to an appropriate position for forwarding the data to the BS. The time required by the UAV to move from the data collection position u_k^{C} to the data forwarding position u_k^{R} can be given by

$$t_{k,i}^{\mathrm{FR}} = \frac{\left\| u_{k,i}^{\mathrm{R}} - u_{k,i}^{\mathrm{C}} \right\|}{v}, \quad k \in \mathcal{K}. \tag{6}$$

The time required by the UAV to forward the input data of the ith task in the kth group to the BS can be given by

$$t_{k,i}^{\mathrm{R}} = \frac{b_{k,i}}{r_{k,i}^{\mathrm{R}}}, \quad k \in \mathcal{K}. \tag{7}$$

In general, the EC embedded in the BS has super computing power. So the time it takes to execute the task is ignored.

Additionally, as the computation results are of small size, the feedback delay from the BS to the UE and from the UAV to the UE is ignored.

4. Problem Formulation

This paper focuses on the multitask scheduling method encountered by the UAV when providing relaying and MEC services to the UE. Assume that the applications of UE are delay sensitive. The objective of the multitask scheduling strategy is to minimize the time required to finish a mobile application issued by the UE. According to the above analysis, the BS has more powerful computation power than the UAV, but it will take much more time for the BS to acquire the input data for processing the task. Therefore, the main consideration of the multitask scheduling strategy is to determine where a task is processed, at the UAV or at the BS?

Firstly, we can divide the tasks belonging to an application into two categories, the tasks in one of the categories to be processed at the UAV and the other at the BS. Therefore, we can simply assume that there are only two big tasks in any kth task-group, i.e., $n_k = 2$. The two big tasks are labelled as i and j ($i \neq j$), respectively. This simplification can be easily extended to the case where each kth task-group has more than two tasks, i.e., $n_k \geq 2$.

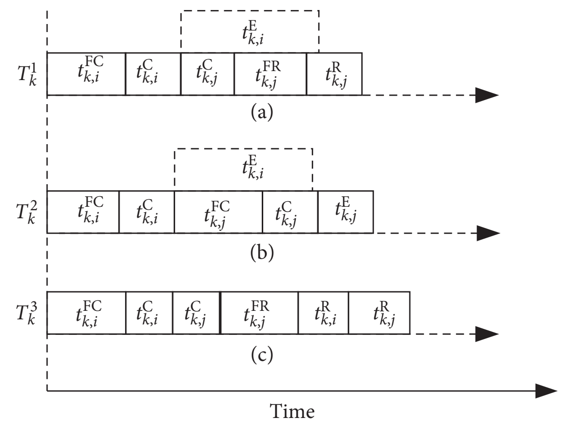

Figure 3: Three cases for applying the proposed scheduling method.

Next, the choice of the execution sites (for the two big tasks in any kth task-group) falls into the following three cases. Figure 3

(1) Case I: one task is executed at the UAV and the other at the BS. This case is shown in Figure 3(a). The time required to complete the kth task-group of the UE can be given by

$$T_k^1 = t_{k,i}^{\mathrm{FC}} + t_{k,i}^{\mathrm{C}} + \max\left(t_{k,i}^{\mathrm{E}}, \left(t_{k,j}^{\mathrm{C}} + t_{k,j}^{\mathrm{FR}} + t_{k,j}^{\mathrm{R}}\right)\right), \quad k \in \mathcal{K}. \tag{8}$$

(2) Case II: both tasks are processed at the UAV. While both of the tasks are with light computation load they can be processed by the UAV independently. This case is illustrated in Figure 3(b). In this case, the time to complete the kth task-group of the UE can be given by

$$T_k^2 = t_{k,i}^{\mathrm{FC}} + t_{k,i}^{\mathrm{C}} + \max\left(t_{k,i}^{\mathrm{E}}, \left(t_{k,j}^{\mathrm{FC}} + t_{k,j}^{\mathrm{C}}\right)\right) + t_{k,j}^{\mathrm{E}}, \quad k \in \mathcal{K}. \tag{9}$$

(3) Case III: both tasks are computed at the BS. When both tasks are very complex, it is preferable to offload them to the BS for computing. In this case, as shown in Figure 3(c), the time required to complete the kth task-group can be given by

$$T_k^3 = t_{k,i}^{\mathrm{FC}} + t_{k,i}^{\mathrm{C}} + t_{k,j}^{\mathrm{C}} + t_{k,i}^{\mathrm{FR}} + t_{k,i}^{\mathrm{R}} + t_{k,j}^{\mathrm{R}}, \quad k \in \mathcal{K}. \tag{10}$$

In order to minimize the completion time of all the tasks of an application, we formulated an optimization problem as

$$\begin{aligned} \min_{U, k \in \mathrm{pre}(a)} \quad & \sum_{k=1}^{K} \min\left(T_k^1, T_k^2, T_k^3\right) \\ \text{s.t.} \quad & v \leq V_{\max}, \\ & u_{1,i}^{\mathrm{I}} = u^{\mathrm{I}}. \end{aligned} \tag{11}$$

where U denotes a set of the UAV's horizontal locations, i.e., the trajectory of the UAV; u^{I} is the initial position of the

UAV during its first task-group; $k \in \mathrm{pre}(a)$ denotes the set of the immediate predecessor tasks of the kth task-group.

This proposed UAV-assisted MEC could accelerate the execution of the computing tasks. The tradeoff of the completion time lies in the fact that although the UAV has much lower computing power than the BS, the BS consumes much more time than the UAV to acquire the task-input data.

5. Algorithmic Design

By inspecting problem (11), the particularity of this problem makes it impossible to use the conventional convex optimization method to solve the problem. Before setting up our algorithm, the following lemma is given first which can simplify our analysis.

Lemma 1. *For any kth task-group, the optimal hovering positions for the UAV to collect the task-input data from the UE and relay data to the BS should satisfy the condition*

$$\begin{aligned} &0 \le x_k^{\mathrm{C}} \le x_k^{\mathrm{R}} \le L, \\ &y_k^{\mathrm{C}} = y_k^{\mathrm{R}} = 0, \\ &k \in \mathrm{K}. \end{aligned} \tag{12}$$

Proof. First, we prove $y_k^{\mathrm{C}} = y_k^{\mathrm{R}} = 0$ for $k \in \mathcal{K}$. For that purpose, we assume that the optimal hovering positions for the UAV are with $y_k^{\mathrm{C}} \ne 0$ and $y_k^{\mathrm{R}} \ne 0$. Then one sets $y_k^{\mathrm{C}} = 0$ and $y_k^{\mathrm{R}} = 0$ and can get the improved channel powers g_k^{C} and g_k^{R} in (1). As the resultant data uploading rates r_k^{C} and r_k^{R} in (2) are also increased, this leads to a smaller value for the objective function of problem (11). It implies that the optimal solution to the problem (11) can be found with $y_k^{\mathrm{C}} = y_k^{\mathrm{R}} = 0$ for $k \in \mathcal{K}$.

Next, we prove that $0 \le x_k^{\mathrm{C}} \le x_k^{\mathrm{R}} \le L$ for $k \in \mathcal{K}$. According to (1) and (2), the UAV can achieve the maximum data rate for collecting the task-input data from the UE at $x_k^{\mathrm{C}} = 0$ and can achieve the maximum data rate for relaying data to the BS at $x_k^{\mathrm{R}} = L$. Note that the depot, i.e., the initial position of the UAV, is between the UE and the BS. It is obvious that if the UAV flies out of the interval $[0, L]$, the flight time increases while the data transmission rate decreases. So, we have $0 \le x_k^{\mathrm{C}} \le x_k^{\mathrm{R}} \le L$ for $k \in \mathcal{K}$.

By applying Lemma 1, g_k^{C} and g_k^{R} in (1) can be simplified to

$$\begin{aligned} g_k^{\mathrm{C}} &= \frac{\beta_0}{\left(x_k^{\mathrm{C}}\right)^2 + h^2}, \\ g_k^{\mathrm{R}} &= \frac{\beta_0}{\left(L - x_k^{\mathrm{R}}\right)^2 + h^2}, \\ &k \in \mathcal{K}. \end{aligned} \tag{13}$$

Based on the above analysis, the trajectory of the UAV during assisting offloading the tasks to the BS can be expressed by

$$\mathbf{U} = \begin{bmatrix} u_{1,i}^{\mathrm{C}} & u_{1,j}^{\mathrm{C}} & u_{1,i}^{\mathrm{R}} & u_{1,j}^{\mathrm{R}} \\ u_{2,i}^{\mathrm{C}} & u_{2,j}^{\mathrm{C}} & u_{2,i}^{\mathrm{R}} & u_{2,j}^{\mathrm{R}} \\ \vdots & \vdots & \vdots & \vdots \\ u_{K,i}^{\mathrm{C}} & u_{K,j}^{\mathrm{C}} & u_{K,i}^{\mathrm{R}} & u_{K,j}^{\mathrm{R}} \end{bmatrix}, \tag{14}$$

where $u_{k,i}^{\mathrm{C}} = u_{k,j}^{\mathrm{C}}, u_{k,i}^{\mathrm{R}} = 0$, and $u_{k+1}^{\mathrm{I}} = u_{k,j}^{\mathrm{R}}$ for $k \in \mathcal{K}$ in Case I. In Case II, we have $u_{k,i}^{\mathrm{R}} = u_{k,j}^{\mathrm{R}} = 0$ and $u_{k+1}^{\mathrm{I}} = u_{k,j}^{\mathrm{C}}$ for $k \in \mathcal{K}$. In Case III, we have $u_{k,i}^{\mathrm{C}} = u_{k,j}^{\mathrm{C}}, u_{k,i}^{\mathrm{R}} = u_{k,j}^{\mathrm{R}}$, and $u_{k+1}^{\mathrm{I}} = u_{k,j}^{\mathrm{R}}$ for $k \in \mathcal{K}$.

Therefore, the original problem (11) can be rewritten as

$$\begin{aligned} \min_{U, k \in \mathrm{pre}(a)} \quad & \sum_{k=1}^{K} \min\left(T_k^1, T_k^2, T_k^3\right) \\ \text{s.t.} \quad & v \le V_{\max}, \\ & u_{1,i}^{\mathrm{I}} = u^{\mathrm{I}}, \\ & 0 \le u_{k+1,i}^{\mathrm{C}} \le u_{k,j}^{\mathrm{R}} \le L, \quad k \in \mathcal{K}. \end{aligned} \tag{15}$$

Although it is difficult to mathematically judge whether problem (15) is convex or not, we can infer that the objective function of the problem is bounded as it is a continuous function over a closed interval. Therefore, we consider a heuristic algorithm based on particle swarm optimization (PSO) [22] and an iterative algorithm to deal with this problem.

5.1. Algorithm Development. A basic PSO algorithm is implemented by a set of candidate solutions (called particles). These particles move in search space according to several simple particle positions and velocity formulas. A particle's motion is affected by its local optimal position, and the particle is also guided to the optimal position in the search space. The optimal position will also help find better positions for the other particles. Repeat this process until the swarm gets the best solution.

According the basic PSO algorithm, for any kth task-group, we take the objective function T ($T = T_k^1, T_k^2$, or T_k^3) as the fitness function. Let Q be the number of particles in the swarm. Each particle is with a position $\mathbf{x}_q = (x_{k,q}^{\mathrm{C}}, x_{k,q}^{\mathrm{R}})$, $q = 1, \ldots, Q$, in the search space $[0, L]$, and a velocity $\mathbf{v}_q$. The fitness function takes a candidate solution x_q in the form of a vector of real numbers and generates a real number as the output. Let μ_q be the best known position of particle Q. Let $\widehat{\mu}$ be the best known position of the entire swarm. The proposed PSO algorithm to search for the optimal positions of the UAV is described in Algorithm 1.

Note that PSO is a heuristic algorithm that makes few or no assumptions about optimization problems and can search very large candidate solution spaces. Heuristics such as PSO algorithm do not absolutely guarantee that an optimal solution is ever found [23], and the convergence capabilities of different PSO algorithms still depend on empirical results and proper control parameter settings [24]. The discussion of PSO-based algorithm designing is beyond the scope of this paper (see [23, 24]).

```
(1) For each particle $q = 1, \ldots, Q$ do:
(a)   Initialize the particle's position with a uniformly distributed random vector $\mathbf{x}_q \sim f(x_{\min}, x_{\max})$, where $f(\cdot)$ is the uniform distribution function, and $x_{\min}$ and $x_{\max}$ are the lower and upper boundaries of the search-space.
(b)   Initialize the $q$th particle's best known position to its initial position: $\mu_q \longleftarrow \mathbf{x}_q$. If $T(\widehat{\mu}) > T(\mu_q)$, update the swarm's best known position: $\widehat{\mu} \longleftarrow \mu_q$.
(c)   Initialize the particle's velocity: $\mathbf{v}_q \longleftarrow f(-b(x_{\max} - x_{\min}), b(x_{\max} - x_{\min}))$, where $b (0 < b < 1)$ is the speed adjustment parameter of the particles.
(2) Until a termination criterion is met (e.g., number of iterations performed or adequate fitness reached), repeat: (1) For each particle $q = 1, \ldots, Q$ do:
(a)   Pick random numbers: $c_1, c_2 \sim f(0, 1)$
(b)   Update the particle's velocity:
      $\mathbf{v}_q \longleftarrow \rho\mathbf{v}_q + w_1 c_1 (\mu_q - \mathbf{x}_q) + w_2 c_2 (\widehat{\mu} - \mathbf{x}_q)$, where $\rho$ is the particle inertia weight, and $w_1$ and $w_2$ are the learning factors.
(c)   Update the particle's position: $\mathbf{x}_q \longleftarrow \mathbf{x}_q + \mathbf{v}_q$
(d)   If $T(\mu_q) > T(\mathbf{x}_q)$, update the particle's best known position: $\mu_q \longleftarrow \mathbf{x}_q$. And, if $T(\widehat{\mu}) > T(\mu_q)$, update the swarm's best known position: $\widehat{\mu} \longleftarrow \mu_q$.
(3) Now $\widehat{\mu}$ holds the best found solution.
```

Algorithm 1: PSO algorithm to obtain the optimal hovering position of the UAV.

```
(1)  **Initialize** $k = 1$, $u^{\mathrm{I}}$, $T = 0$
(2)  **Repeat**
(3)    Obtain the optimal solution of $T_k^1$, $T_k^2$ and $T_k^3$ by Algorithm 1;
(4)    Find the optimal scheduling mode, $T_k = \min(T_k^1, T_k^2, T_k^3)$;
(5)    Update the initial position of the UAV for the next task-group:
(6)      **if** $T_k = T_k^1$ or $T_k = T_k^3$ **then**
(7)        $u_{k+1}^{\mathrm{I}} = u_{k,j}^{\mathrm{R}}$;
(8)      else
(9)        $u_{k+1}^{\mathrm{I}} = u_{k,j}^{\mathrm{C}}$;
(10)   Update the time $T$ for completing the previous $k$ task-groups: $T = T + T_k$;
(11)   Update $k = k + 1$;
(12) **Until** a maximum number of iterations has been reached (i.e., $k > K$)
(13) Now $T$ is the minimum time to complete all the tasks.
```

Algorithm 2: An iterative algorithm for problem (15).

By using the proposed PSO, an iterative algorithm to solve problem (15) is summarized in the following Algorithm 2.

6. Simulation Results

In this section, we provide the simulation results to verify the performance of the proposed multitask scheduling method. The simulated system has one UE and one BS as shown in Figure 1. The distance between the UE and the BS is $L = 1000$ m and there is no direct link between them due to obstruction. We set the communication bandwidth of the system as $B = 10$ MHz, the channel power gain as -30 dB at the reference distance of 1 m, and the noise power as 10^{-9} W at the UE. The maximum horizontal flying speed of the UAV is set as $V = 30$ m/s. The flying altitude of the UAV is fixed at $H = 40$ m. The transmission powers at the UE and at the UAV are set as 20 dBm and 25 dBm, respectively. The maximum computing capacity of the UAV is set as 2 GB/s and the number of CPU cycles required to compute each bit is set as 300 cycles/bit. The coordinates of the UE and the BS are fixed at $(0, 0, 0)$ and $(0, 0, L)$, respectively. We assume that the initial position of the UAV at the beginning of executing the tasks of the UE is $u^{\mathrm{I}} = (L/2, 0, H)$.

Firstly, we show the convergence behavior of the PSO Algorithm 1 and illustrate the convergence of the proposed iteration Algorithm 2. The convergence behavior of the Algorithm 1 for three scheduling methods (i.e., Case I, Case II, and Case III) is shown in Figure 4, where the size of the App data is 100 MB (the task i is 30 MB and the task j is 70 MB) and the initial position of the UAV is $u^{\mathrm{I}}=(L/2, 0, H)$. It is observed that the fitness value obtained by the PSO Algorithm in three cases decreases quickly with the number of iterations. Case I converges at 5 iterations, and Case II and Case III converge at 15 iterations. The proposed Algorithm 2 obtains the optimal scheduling method by K-times iterative the PSO Algorithm, so its convergence depends on the PSO Algorithm. It can be seen from Figure 4 that the PSO Algorithm only needs several iterations to converge, so Algorithm 2 is also convergent, and its time complexity is K times that of the PSO Algorithm.

Next, to demonstrate the performance of the proposed scheduling method, the following 3 baseline scheduling methods are also simulated for comparison purpose.

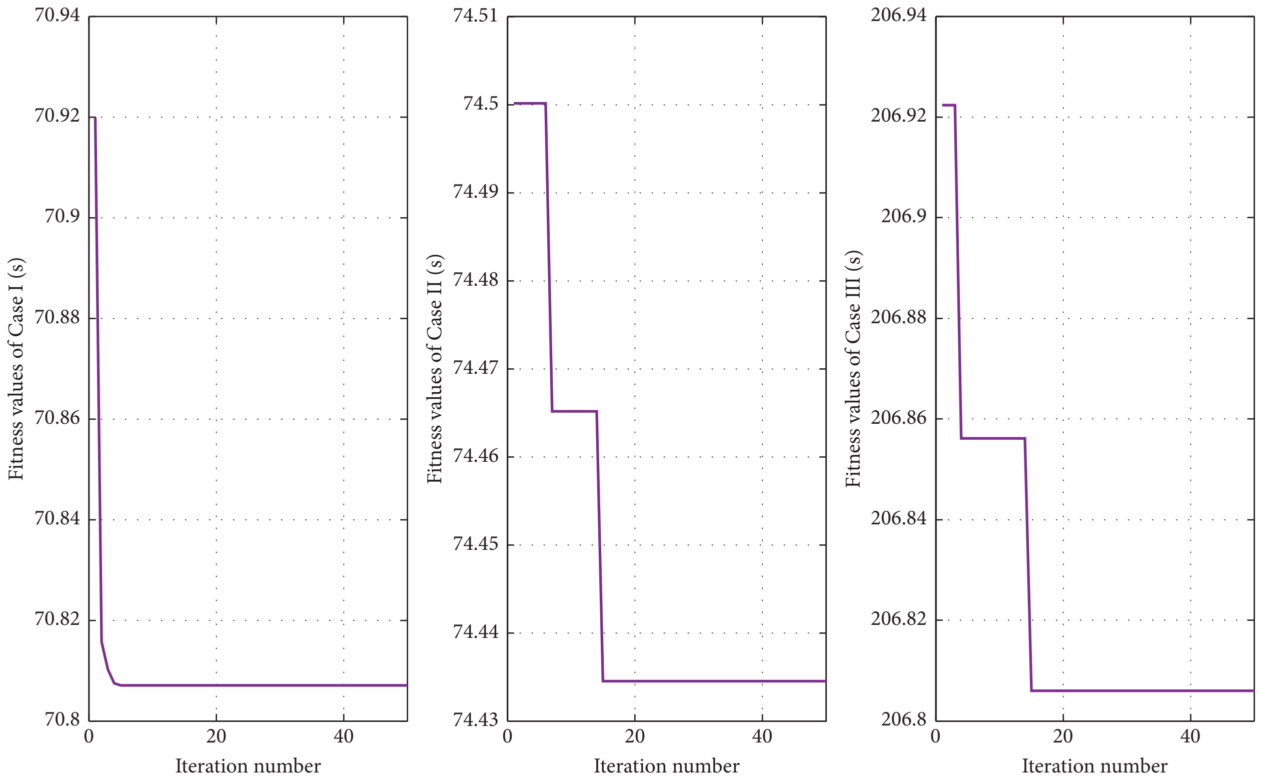

FIGURE 4: Convergence behavior of the PSO algorithm.

6.1. Stationary Scheduling Method. In this method, the position of the UAV is always fixed at the initial position $(L/2, 0, H)$. Firstly, the UAV collects the data bits of one task of the UE. Next, the UE should decide whether to perform the task locally or offload it to the BS depending on how long the task is completed. The tasks of the UE are performed one after another until all of them are completed. This method considers an extreme case where the UAV minimizes the flight time but does not get good channels for task offloading.

6.2. Cyclic Scheduling Method. The UAV first flies to and hovers right above the UE (thus having the best channel condition) and starts to collect the data bits of the first task of the UE. After that, the UAV will immediately execute the first task and, at the same time, start to collect the data bits of the second task of the UE. After collecting the data bits of the second task, the UAV will fly to and hover right above the BS and offload the data bits to the BS. This method considers another extreme case where the UAV can get the best channel for task offloading but at the expense of extended flight time.

6.3. UAV Execution Only Method. In this method, the UAV flies from the initial position and hovers over the UE. All the collected tasks are executed by the UAV without the participation of the BS.

We show the performance of the proposed multitask scheduling method when tackling mobile applications with different sizes. For any size of the applications, it is divided into four task-groups. The size of each task-group is randomly assigned. Each task-group includes two tasks i and j and the sizes of i and j are in a ratio of $1:2$. In Figure 5, we show the completion time versus different application sizes by applying our proposed and three other task scheduling methods.

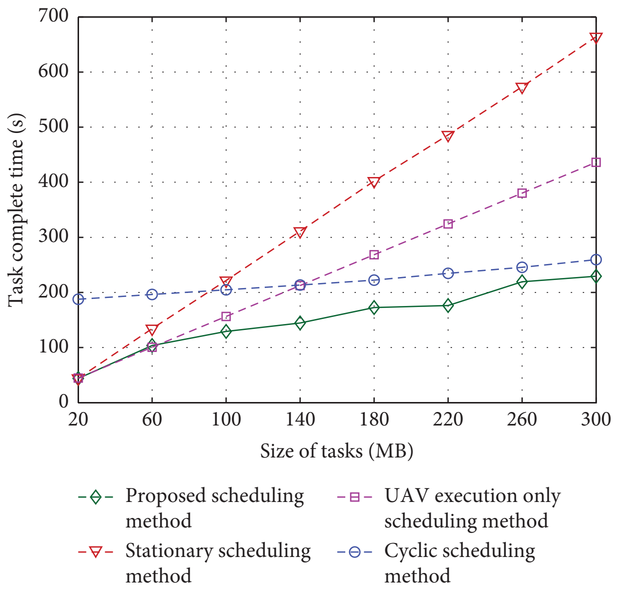

FIGURE 5: The completion time for the applications with different sizes.

From Figure 5, we can observe that the proposed method outperforms the other three methods for any size application. It is also noted that the performance gap between the proposed method and the cyclic scheduling method decreases with the increasing application size. This indicates that when the size of an application becomes large, the proposed task scheduling method could guide the UAV to fly closer to the UE (or the BS) for collecting data (or offloading data), which reduces the time for finishing all the tasks of the application. This also indicates that when the size of an application becomes large, the system tends to fall into Case I and Case III as introduced in Section 4.

In order to better understand how the UAV help the UE perform the mobile applications, we show the flight trajectory and the hovering positions of the UAV. In the simulation, we set the size of the application as 140 MB, and it is divided into four task-groups. The size of each task-group is randomly assigned but the sizes of the two tasks in a group are still in a ratio of 1 : 2. Since y-axis in the UAV is fixed, we only show in Figure 6 the change of x-axis coordinates of the UAV during the task execution.

The solid line in Figure 6 represents the flight trajectory of the UAV to execute the two tasks in the same task-group, the dotted line represents the flight trajectory of the UAV across two consecutive task-groups, and the arrow indicates the flight direction of the UAV. In the simulation, we note that except for executing the first task-group our proposed method falls into Case II; however, it all falls into Case I when executing the other three task-groups. This is due to the relatively small size of the tasks in the first task-group. In addition, we also note that when executing the second task-group, there is only one hover point for the UAV. This is due to the small size of the second task-group, and the UAV would like to offload the input data of the second task to the BS immediately after collecting the data from the UE.

We testify the influence of the number of task-groups on the completion time. For that purpose, we set the size of an application as 300 MB and the application can be separated into multiple task-groups. By using the different task scheduling methods, the completion time versus different number of task-groups is shown in Figure 7.

In Figure 7, we can observe that the number of task-groups that an application is divided into directly affects the completion time of the application. By using the stationary scheduling method and the UAV calculation only method, the task completion time does not change with the increasing number of the task-groups. This is because the UAV is stationary and the number of the task-groups does not increase the flight time of the UAV. It is noted that when the number of the task-groups is greater than 7 the task completion time will decrease. As an application is divided into more task-groups, the size of each task will decrease. The UAV can thus adopt the scheduling method in Case II to reduce the execution time of the application.

We show the flight trajectory and the hovering positions of the UAV in Figure 8. The size of the application is 300 MB and the application is divided into 4 task-groups. In the simulation to generate Figure 8, we note that the scheduling of the 1th and 3rd task-group falls into Case I, the scheduling of 2nd task-group falls into Case II, and the scheduling of the 4th task-group falls into Case III. Then we can draw a conclusion that the proposed multitask scheduling method can always find the best tradeoff between the UAV location and the channel condition. Thus, in comparison to the other scheduling methods, it can use the minimum time to complete a computation-intensive mobile application.

Finally, to illustrate the effect of the speed of the UAV on the proposed scheduling method, we assume that an application size of the UE is 100 MB, and the application is divided into three task-groups, each with two tasks i and j, as shown in Figure 9. The solid line in the graph indicates the

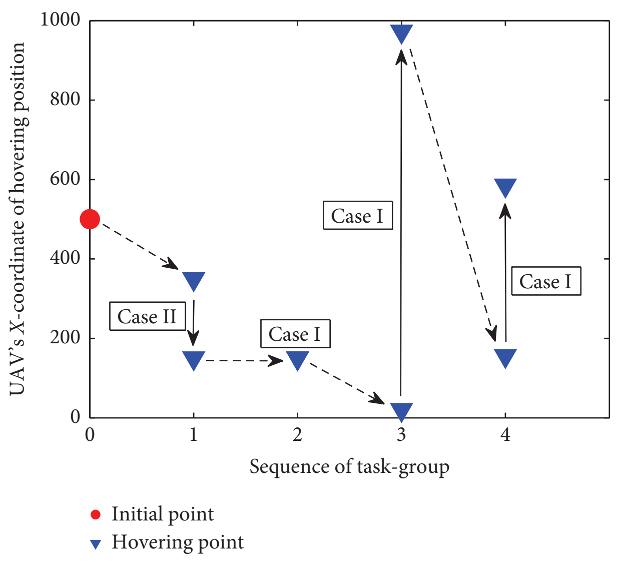

Figure 6: The trajectory of the UAV with 150 MB size of the app.

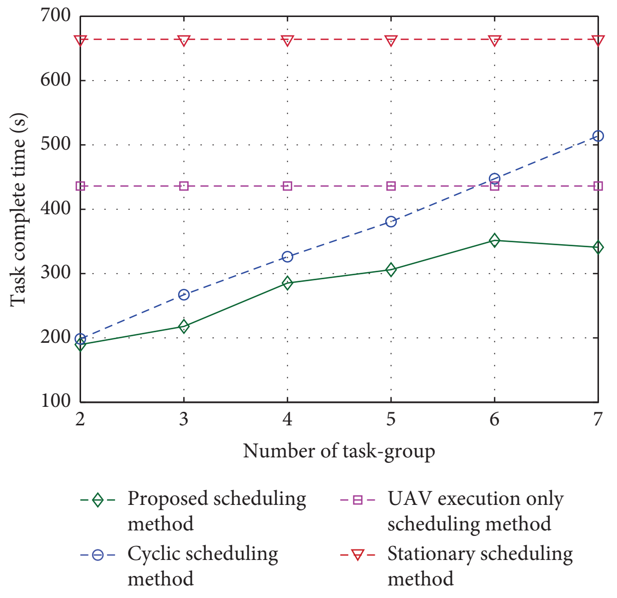

Figure 7: The completion time for an application with different number of task-groups.

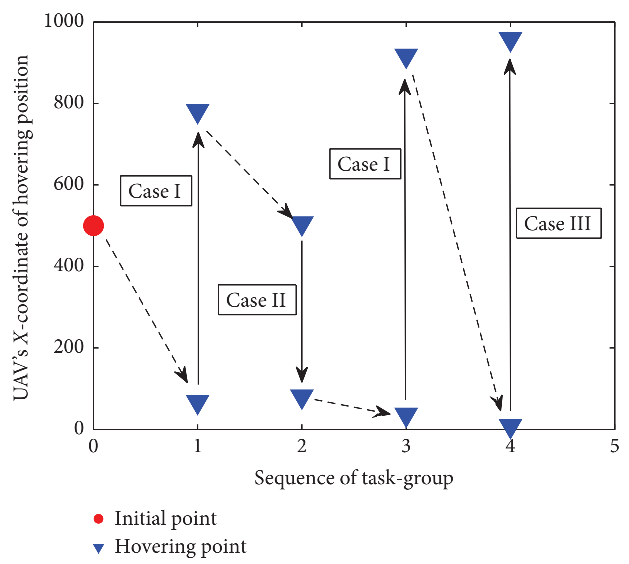

Figure 8: The trajectory of the UAV with 300 MB size of the app.

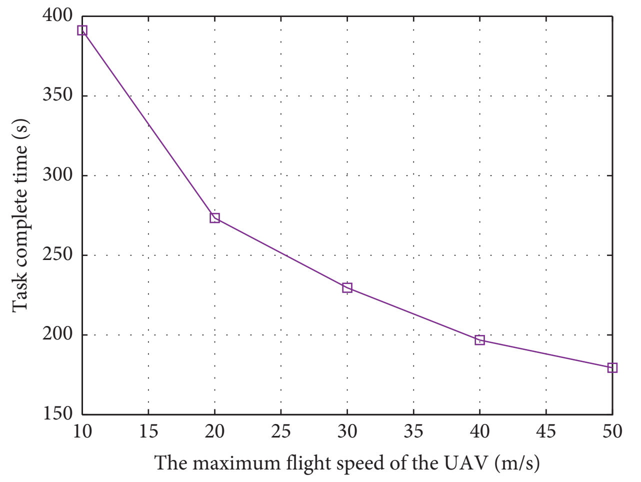

Figure 10: Task completion time at different flight speed of the UAV.

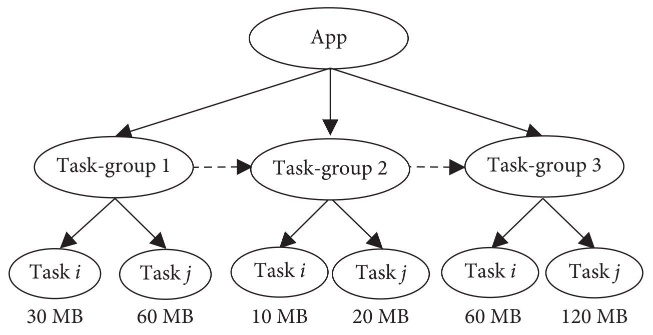

Figure 9: An example of task partitioning with 150 MB size of the app.

Table 1: Scheduling methods for different UAV's speeds.

The UAV's speed (m/s)	Task-group 1	Task-group 2	Task-group 3
10	Case I	Case II	Case I
20	Case I	Case II	Case I
30	Case I	Case II	Case III
40	Case I	Case I	Case III
50	Case I	Case I	Case III

task division, and the dashed line indicates the priority order of the task-groups. It can be seen from Figure 9 that the size of the three task-groups is 90 MB, 30 MB, and 180 MB, respectively. The size of the task j in each task-group is twice that of the task i. For different UAV speeds, the scheduling methods selected by the three task-groups are shown in Table 1.

It can be seen in Table 1 that the task-groups intelligently choose the appropriate scheduling method with the increase of the UAV speed. For task-group 1, the scheduling method does not change with the increase of UAV speed. This is because the tasks are relatively medium-sized tasks in task-group 1, so Case I is optimal at any speed. For task-group 2, it can be seen from Figure 9 that both tasks are relatively small. When the UAV's flight speed is lower than 30 m/s, the UAV's flight time overhead will exceed the time calculated by the UAV. It is obviously not cost-effective to select Case I and Case III. When the speed of the UAV is more than 40 m/s, Case I becomes the optimal scheduling method because of the increase of flight speed. For task-group 3, with the increase of the UAV speed, the optimal scheduling method changes from Case I to Case III. This is because that the UAV has a long flight time due to its low flight speed, and there is enough time to calculate the task i on the UAV. When the UAV speed is greater than 30 m/s, it is obviously appropriate to offload two large-scale tasks (e.g., task i and task j) to the BS.

In Figure 10, the task completion time is plotted versus the different flight speed of the UAV. Obviously, task completion time decreases significantly with the increase of the UAV speed. It is worth noting that the decreasing trend of task completion time in Figure 10 is gradually flattening out. According to Lemma 1, the hover position of the UAV must be between the UE and BS, and the location of the UE and the BS is fixed. Therefore, the speed of the UAV is limited to improve the performance of the task completion time. When the speed of the UAV is big enough, it becomes a secondary factor affecting the task completion time.

7. Conclusions and the Future Work

This paper studied the UAV-assisted MEC model where the computation tasks of the UE have strict order dependencies. A PSO algorithm as well as an iterative algorithm was developed to minimize the completion time of the offloaded computation tasks. Simulation results show that the proposed multitask scheduling method can always find the best tradeoff between the UAV's position and the channel condition. However, this paper considers that the UAV can only receive data while hovering at a position. In the future, we will study the case where the UAV can also receive data while flying.

In addition, only one UE is considered in the system model of this paper, but this provides us with ideas and insights into studying multi-UE. Although this paper considers that the UAV-assisted MEC model only involves one UE, the scheduling method for the one-UE case could provide us the insights into studying the multi-UE case in the future. On the basis of the research results of this paper, we will systematically discuss the multi-UE situation in future research work, including the optimal hovering position and flight path of the UAV, the selection of offloading position, and task scheduling method. In terms of the UAV flight path, we consider that it is no longer a simple straight-line trajectory. Since we assume that the data can only be received or transmitted when the UAV is hovering, there will be multiple optimal hovering positions within the area covered by the UEs. The UAV could choose the optimal flight path according to the optimal hovering position. In terms of the task scheduling method, at any hovering point, the UAV can serve multiple UEs, and each UE can select the optimal time instant to offload the tasks. Considering the priority relationship of the tasks of an application, the UEs can only offload the data of one task-group at a time. Then they can continue to offloading the data of the next task-group after the result is obtained.

Data Availability

The simulation codes data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant nos. 61971421, 61620106011, and U1705263) and Zhongshan City Team Project (Grant no. 180809162197874).

References

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the NIPS*, Montreal, Canada, 2014.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the NIPS*, Lake Tahoe, NV, USA, 2012.

[3] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the Internet of vehicles: offloading framework and job scheduling," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 28–36, 2019.

[4] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660–10675, 2017.

[5] Z. Dawy, W. Saad, A. Ghosh, J. G. Andrews, and E. Yaacoub, "To-wards massive machine type cellular communications," *IEEE Wireless Commun*, vol. 24, no. 1, 2017.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[7] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[8] K. Wang, K. Yang, H.-H. Chen, and L. Zhang, "Computation diversity in emerging networking paradigms," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 88–94, 2017.

[9] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," in *Proceedings of the GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, Singapore, December 2017.

[10] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 4983–4996, 2016.

[11] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7574–7589, 2017.

[12] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "UAV-enabled mobile edge computing: offloading optimization and trajectory design," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[13] Y. Du, K. Yang, K. Wang, G. Zhang, Y. Zhao, and D. Chen, "Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IOT systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10187–10200, 2019.

[14] J. Hu, M. Jiang, Q. Zhang, Q. Li, and J. Qin, "Joint optimization of UAV position, time slot allocation, and computation task partition in multiuser aerial mobile-edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7231–7235, 2019.

[15] Y. Wang, M. Hua, Z. Liu, D. Zhang, B. Ji, and D. Haibo, "UAV-based mobile wireless power transfer systems with joint optimization of user scheduling and trajectory," *Mobile Networks and Applications*, vol. 4, pp. 1–15, 2019.

[16] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[17] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 401–411, ACM, Singapore, 2018.

[18] S. Wang, M. Xia, K. Huang, and Y.-C. Wu, "Wirelessly powered two-way communication with nonlinear energy harvesting model: rate regions under fixed and mobile relay," *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8190–8204, 2017.

[19] C. Tang, M. Hao, X. Wei, and W. Chen, "Energy-aware task scheduling in mobile cloud computing," *Distributed and Parallel Databases*, vol. 36, no. 3, pp. 529–553, 2018.

[20] Y. Zeng, J. Lyu, and R. Zhang, "Cellular-connected UAV: potential challenges and promising technologies," *IEEE Wireless Commun*, vol. 26, no. 1, 2019.

[21] Q. Wu, L. Liu, and R. Zhang, "Fundamental trade-offs in communication and trajectory design for UAV-enabled wireless network," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 36–44, 2019.

[22] G. Zhang, K. Yang, P. Liu, E. Ding, and Y. Zong, "Joint channel bandwidth and power allocation game for selfish cooperative relaying networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 9, pp. 4142–4156, 2012.

[23] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[24] M. E. H. Pedersen, "Good parameters for particle swarm optimization," Tech. Rep. HL1001, Hvass Lab, Copenhagen, Denmark, 2010.