Joint Deployment and Task Scheduling Optimization for Large-Scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing

Yong Wang, *Senior Member, IEEE*, Zhi-Yang Ru, Kezhi Wang, *Member, IEEE*, and Pei-Qiu Huang

Abstract*—This article establishes a new multiunmanned aerial vehicle (multi-UAV)-enabled mobile edge computing (MEC) system, where a number of unmanned aerial vehicles (UAVs) are deployed as flying edge clouds for large-scale mobile users. In this system, we need to optimize the deployment of UAVs, by considering their number and locations. At the same time, to provide good services for all mobile users, it is necessary to optimize task scheduling. Specifically, for each mobile user, we need to determine whether its task is executed locally or on a UAV (i.e., offloading decision), and how many resources should be allocated (i.e., resource allocation). This article presents a two-layer optimization method for jointly optimizing the deployment of UAVs and task scheduling, with the aim of minimizing system energy consumption. By analyzing this system, we obtain the following property: the number of UAVs should be as small as possible under the condition that all tasks can be completed. Based on this property, in the upper layer, we propose a differential evolution algorithm with an elimination operator to optimize the deployment of UAVs, in which each individual represents a UAV's location and the entire population represents an entire deployment of UAVs. During the evolution, we first determine the maximum number of UAVs. Subsequently, the elimination operator gradually reduces the number of UAVs until at least one task cannot be executed under delay constraints. This process achieves an adaptive adjustment of the number of UAVs. In the lower layer, based on the given deployment of UAVs, we transform the task scheduling into a 0-1 integer programming problem. Due to the large-scale characteristic of this 0-1 integer programming problem, we propose an efficient greedy algorithm to obtain the near-optimal solution with much less time. The effectiveness of the proposed two-layer optimization method and the established multi-UAV-enabled MEC system is demonstrated on ten instances with up to 1000 mobile users.

***Index Terms*—Deployment, differential evolution (DE), mobile edge computing (MEC), multiunmanned aerial vehicle (multi-UAV), task scheduling, two-layer optimization.**

Manuscript received January 5, 2019; revised May 10, 2019; accepted August 12, 2019. This work was supported in part by the Innovation-Driven Plan in Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grant 61673397 and Grant 61976225, in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant 2016JJ1018, and in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06. This article was recommended by Associate Editor M. Zhang. *(Corresponding author: Kezhi Wang.)*

Y. Wang, Z.-Y. Ru, and P.-Q. Huang are with the School of Automation, Central South University, Changsha 410083, China (e-mail: ywang@csu.edu.cn; zhiyang.ru@csu.edu.cn; pqhuang@csu.edu.cn).

K. Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: kezhi.wang@northumbria.ac.uk).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2935466

I. Introduction

With the increasing popularity of mobile devices, more and more new types of mobile applications have emerged, such as mobile online gaming [1] and speech recognition [2]. However, these applications are sensitive to latency and require considerable computational resources. Due to the physical limitations, such as battery power and computation resources, it poses a great challenge for mobile devices to execute these applications [3].

Mobile edge computing (MEC), which deploys servers to the network edge [4], [5], has been considered as a promising technology to address this challenge. In MEC, mobile devices can offload their tasks to the servers close to them. Compared with mobile cloud computing, MEC consumes less transmission time and energy due to a shorter transmission distance. However, the locations of MEC servers are usually fixed and cannot be flexibly changed according to the needs of mobile users, which limits the MEC's capability.

In recent years, unmanned aerial vehicles (UAVs) have received extensive attention in wireless communications [6]–[8]. For example, UAVs have been used in areas with limited communication infrastructures, such as in developing countries or mountainous areas, as well as in earthquake response, emergency rescue, and battlefield communication [9]. Very recently, a UAV-enabled MEC wireless-powered system has been studied in [10], in which an MEC server is mounted on a UAV (i.e., a flying edge cloud). This kind of system can provide two advantages: 1) due to the higher altitude, the flying edge cloud can provide better line-of-sight links to mobile users with a higher probability and 2) since the UAV can be flexibly deployed, it can further shorten the transmission distance. Overall, this kind of system can provide better services to mobile users. Therefore, the use of UAVs is expected to play an important role in improving the performance of MEC.

However, the current study in [10] only considers one UAV. A question which arises naturally is whether we can deploy multiple UAVs simultaneously to serve mobile users. Compared with a single UAV, multiple UAVs can support more tasks within a shorter time, which can remarkably boost the applications of MEC in emergency and complicated scenarios. To this end, we make the first attempt to investigate a new multi-UAV-enabled MEC system, where multiple UAVs are employed to serve large-scale mobile users on the ground in a given area. To minimize this system's energy consumption while meeting the needs of all mobile users, there exist two key issues to be addressed: 1) the deployment of UAVs and 2) task scheduling. Specifically, the purpose of the deployment of UAVs is to determine the number and locations of UAVs. In addition, task scheduling includes two aspects: 1) the offloading decision and 2) resource allocation. The former aims at determining whether a task is executed locally or is offloaded to a UAV. Subsequently, the latter decides how many resources should be allocated to this task.

Actually, the deployment of a single UAV/multiple UAVs and the task scheduling in MEC have been extensively studied individually in wireless communications. Next, we briefly introduce them.

1) *Deployment of a Single UAV/Multiple UAVs:* Fan *et al.* [11] researched the node placement of a UAV relaying system, with the aim of maximizing the system throughput. Bor-Yaliniz *et al.* [12] optimized the placement of a UAV to maximize the revenue of the network. Mozaffari *et al.* [13] designed the efficient deployment of multiple UAVs as wireless base stations, in which the total coverage area and the coverage lifetime of UAVs are maximized. Mozaffari *et al.* [14] investigated the placement of UAVs for data collection from ground Internet of Things devices. Lyu *et al.* [15] presented the placement of UAVs to supply distributed ground terminals with wireless coverage, ensuring that each ground terminal can be served by at least one UAV. Sharma *et al.* [16] introduced the assignment of UAVs over geographical areas to meet high traffic demands. Mozaffari *et al.* [17] deployed a UAV as a flying base station to provide wireless communications to an area.
2) *Task Scheduling:* Some researchers have focused on either the offloading decision or the resource allocation in task scheduling of MEC. For example, Zhang *et al.* [18] proposed an energy-efficient offloading decision mechanism for MEC in 5G heterogeneous networks. Lyu *et al.* [19] designed a selective offloading decision scheme in MEC to minimize the energy consumption of Internet of Things devices. Wang *et al.* [20] optimized the resource allocation in MEC by means of a unifying framework for the power-performance tradeoff of a mobile service provider. You *et al.* [21] investigated the resource allocation for a multiuser MEC system based on time-division multiple access and orthogonal frequency-division multiple access. Recently, much attention has been paid to optimize the offloading decision and resource allocation in MEC simultaneously. For instance, Mao *et al.* [22] presented an effective computation offloading strategy for a green MEC system with energy harvesting devices by optimizing the offloading decision and the resource allocation simultaneously. Zhang *et al.* [23] suggested the simultaneous offloading decision and resource allocation optimization in MEC to minimize the energy consumption and monetary cost from the mobile terminals' perspective. Kan *et al.* [24] introduced the offloading decision and the resource allocation of the MEC server considering the variety of tasks' requirements.

From this introduction, it is clear that the joint optimization of the deployment of UAVs and task scheduling remains scarce in current studies. Moreover, in MEC, large-scale mobile users have rarely been taken into consideration. Due to the fact that the system developed in this article involves both multi-UAV-enabled MEC and mobile users, we must jointly optimize the deployment of UAVs and task scheduling. To the best of our knowledge, this article is the first attempt to investigate joint deployment and task scheduling optimization for large-scale mobile users in a multi-UAV-enabled MEC system.

The main contributions of this article are summarized as follows.

1) A new multi-UAV-enabled MEC system is proposed, where multiple UAVs are used as flying edge clouds for large-scale mobile users. This system can further develop the capability of traditional MEC systems by using multiple UAVs.
2) A two-layer optimization method called ToDeTaS is proposed to jointly optimize the deployment of UAVs and task scheduling, with the purpose of minimizing the system energy consumption. Specifically, we optimize four aspects: the number and locations of UAVs, the offloading decision, and the resource allocation.
3) In the upper layer, a differential evolution (DE) algorithm with an elimination operator is presented to optimize the deployment of UAVs. We encode a UAV's location into an individual and the entire population represents an entire deployment of UAVs. After analyzing this system, to achieve the minimum energy consumption, we should give a priority to the number of UAVs under the condition that all tasks can be completed. Based on this property, we first determine the maximum number of UAVs, and gradually reduce the number by the elimination operator if all tasks can be completed. In principle, the number of UAVs is adaptively adjusted by the elimination operator and the locations of UAVs are optimized by DE.
4) With respect to a given deployment of UAVs in the upper layer, the task scheduling in the lower layer is transformed into a 0-1 integer programming problem. To reduce the computational time for the large-scale 0-1 integer programming problem, an efficient greedy algorithm is proposed to obtain the near-optimal solution.
5) Extensive experiments have been carried out on ten instances with up to 1000 mobile users. The experimental results demonstrate the effectiveness of ToDeTaS and the multi-UAV-enabled MEC system.

The remainder of this article is organized as follows. Section II introduces the model and problem formulation of

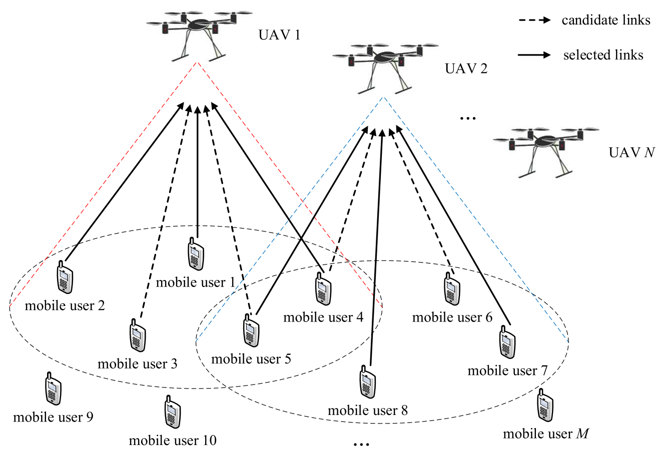


Fig. 1. Multi-UAV-enabled MEC system consisting of M mobile users and N UAVs. As shown in this figure, the tasks of mobile users 1, 2, and 4 are executed on UAV 1; the tasks of mobile users 5, 7, and 8 are executed on UAV 2; and the remaining tasks are executed locally.

the proposed system. Section III describes the details of our proposed ToDeTaS. Section IV gives the experimental studies. Section V discusses two issues. Finally, Section VI concludes this article.

II. System Model and Problem Formulation

As shown in Fig. 1, we consider a multi-UAV-enabled MEC system consisting of M mobile users denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$ and N UAVs denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$. In this system, $(x_i, y_i, 0)$ is the 3-D coordinate of mobile user i ($i \in \mathcal{M}$). In addition, we assume that each mobile user i has a task U_i to be executed. Specifically, U_i can be described as $U_i = (C_i, D_i)$, where C_i describes the total number of the CPU cycles for completing U_i, and D_i denotes the size of input data of mobile user i. Note that M, x_i, y_i, C_i, and D_i can be known *a priori*. As for N UAVs, we assume that they are equipped with directional antennas of fixed beamwidth θ. These UAVs are flying at a constant altitude H and the location of UAV j ($j \in \mathcal{N}$) is represented by (X_j, Y_j, H). It is worth noting that N, X_j, and Y_j cannot be obtained in advance.

In this system, UAVs are used as flying edge clouds. Therefore, each task can be executed on its own mobile device or one of UAVs. As a result, each task has $(N+1)$ execution patterns denoted as $\mathcal{K} = \{0, 1, \ldots, N\}$. Specifically, $k = 0$ ($k \in \mathcal{K}$) indicates that a task is executed on its own mobile device and $k > 0$ indicates that a task is executed on UAV k. Furthermore, we assume that N UAVs serve all mobile users via frequency-division multiple access with an equal bandwidth allocation. In this article, we define matrix $\mathbf{a}$ to denote the offloading decision, where $a_{i,k} = 1$ ($i \in \mathcal{M}$ and $k \in \mathcal{K}$) if U_i is executed in pattern k; otherwise, $a_{i,k} = 0$. For example, in Fig. 1, U_1, U_2, and U_4 are executed on UAV 1; U_5, U_7, and U_8 are executed on UAV 2; and the remaining tasks are executed locally. As a result, $a_{1,1}$, $a_{2,1}$, $a_{4,1}$, $a_{5,2}$, $a_{7,2}$, $a_{8,2}$, $a_{3,0}$, $a_{6,0}$, $a_{9,0}$, $a_{10,0}$, and $a_{M,0} = 1$, and the remaining is equal to zero. In addition, we define another matrix $\mathbf{f}$ to denote the resource allocation, where $f_{i,k}$ ($i \in \mathcal{M}$ and $k \in \mathcal{K}$) is the computation resources allocated to U_i in pattern k.

In our system, there are three models: 1) the local execution model; 2) the MEC execution model; and 3) the UAV hover model.

A. Local Execution Model

When U_i is executed on its own mobile device, the time spent to complete it is defined as [25]

$$T_{i,0}^{L} = \frac{C_i}{f_{i,0}}, \ \forall i \in \mathcal{M}. \tag{1}$$

In addition, the energy used to complete U_i is given as [26]

$$E_{i,0}^{L} = \eta_1 (f_{i,0})^{v-1} C_i, \ \forall i \in \mathcal{M} \tag{2}$$

where η_1 is the effective switched capacitance and v is a positive constant.

B. MEC Execution Model

When a task is executed on a UAV, this task is first transmitted to the UAV, and then it is executed by the MEC server on the UAV. After execution, the result is returned to the mobile user.

For mobile user i, its horizontal distance to UAV j is given as

$$d_{i,j}^{MU} = \sqrt{(x_i - X_j)^2 + (y_i - Y_j)^2} \quad \forall i \in \mathcal{M}, \ j \in \mathcal{N}. \tag{3}$$

Obviously, if U_i is executed on UAV j, mobile user i must be within the coverage area of UAV j. That is, the following constraint should be satisfied [27]:

$$C1: a_{i,k} d_{i,j}^{MU} \le R, \ \forall i \in \mathcal{M}, \ j \in \mathcal{N}, \ k = j \tag{4}$$

where R is the coverage radius of each UAV and $R = H \tan\theta$.

The distance between two UAVs is expressed as

$$d_{j_1,j_2}^{UU} = \sqrt{\left(X_{j_1} - X_{j_2}\right)^2 + \left(Y_{j_1} - Y_{j_2}\right)^2} \quad \forall j_1, \ j_2 \in \mathcal{N}, \ j_1 \ne j_2. \tag{5}$$

Note that any two UAVs must maintain a minimum distance d_{min}^{UU} to avoid collision; thus, another constraint holds [28]

$$C2: d_{j_1,j_2}^{UU} \ge d_{min}^{UU}, \forall j_1, \ j_2 \in \mathcal{N}, \ j_1 \ne j_2. \tag{6}$$

Due to the computational capability limitations of an MEC server, each UAV can only execute at most n_{max} tasks. That is [29]

$$C3: \sum_{i=1}^{M} a_{i,k} \le n_{max}, \ \forall i \in \mathcal{M}, \ k \in \mathcal{K} \backslash \{0\}. \tag{7}$$

The uplink data rate of U_i in pattern k is given as [30]

$$r_{i,k} = B \log_2 \left(1 + \frac{P \beta_0 G_0}{N_0 B \theta^2 \left(\left(d_{i,j}^{MU} \right)^2 + H^2 \right)} \right) \quad \forall i \in \mathcal{M}, \ j \in \mathcal{N}, \ k = j, \tag{8}$$

where B is the channel bandwidth, P denotes the transmission power of each mobile device, β_0 is the channel power gain at the reference distance, G_0 is a positive constant, and N_0 is the noise power spectrum density.

Then, the total time for completing U_i includes the transmission time and the computational time on UAV j [31]

$$T_{i,k}^M = \frac{D_i}{r_{i,k}} + \frac{C_i}{f_{i,k}}, \ \forall i \in \mathcal{M}, \ k = j. \tag{9}$$

In addition, the total energy used to complete U_i includes the transmission energy and the computation energy on UAV j [32]

$$E_{i,k}^M = P\frac{D_i}{r_{i,k}} + \eta_2 (f_{i,k})^{v-1} C_i, \ \forall i \in \mathcal{M}, \ k = j \tag{10}$$

where η_2 is the effective switched capacitance.

Similar to [29], we assume that the output of the task can be returned to the mobile user with negligible transmission delay.

C. UAV Hover Model

When a UAV hovers at its fixed location for some time, the energy for it to hover is expressed as

$$E^H = P_0 T \tag{11}$$

where P_0 and T are the hover power and the hover time, respectively.

Considering that this system contains both multi-UAV-enabled MEC and mobile users, we need to jointly optimize the deployment of UAVs and the task scheduling to minimize the system energy consumption, which includes the energy to complete all tasks in the local computation patten or the MEC computation patten and the energy for UAVs' hover. The joint deployment and task scheduling optimization problem can be formulated as

$$\begin{aligned}
\min_{N, X_j, Y_j, a_{i,k}, f_{i,k}} \quad & \sum_{i=1}^{M} \left(a_{i,0} E_{i,0}^L + \sum_{k=1}^{N} a_{i,k} E_{i,k}^M \right) + \beta N E^H \\
\text{s.t.} \quad & C1: a_{i,k} d_{i,j}^{MU} \le R \\
& \forall i \in \mathcal{M}, \ j \in \mathcal{N}, \ k = j \\
& C2: d_{j_1,j_2}^{UU} \ge d_{min}^{UU}, \forall j_1, \ j_2 \in \mathcal{N}, \ j_1 \ne j_2 \\
& C3: \sum_{i=1}^{M} a_{i,k} \le n_{max} \\
& \forall i \in \mathcal{M}, \ k \in \mathcal{K} \backslash \{0\} \\
& C4: \sum_{k=0}^{N} a_{i,k} = 1, \ \forall i \in \mathcal{M}, \ k \in \mathcal{K} \\
& C5: f_{i,k} > 0, \ \forall a_{i,k} = 1, \ i \in \mathcal{M}, \ k \in \mathcal{K} \\
& C6: f_{i,k} = 0, \ \forall a_{i,k} = 0, \ i \in \mathcal{M}, \ k \in \mathcal{K} \\
& C7: a_{i,0} T_{i,0}^L \le T, \ \forall i \in \mathcal{M} \\
& C8: a_{i,k} T_{i,k}^M \le T, \ \forall i \in \mathcal{M}, \ k \in \mathcal{K} \backslash \{0\}
\end{aligned} \tag{12}$$

where $C4$ ensures that all tasks are executed and each task can only be executed in one pattern; $C5$ and $C6$ denote that if U_i is executed in pattern k, $f_{i,k}$ is greater than 0; otherwise, it is equal to 0; $C7$ and $C8$ are delay constraints for each task; and β is a weight coefficient and set to 1 in this article.

III. Proposed Approach

A. Motivation

From the introduction in Section II, it is clear that (12) is a nonconvex nonlinear optimization problem. Therefore, traditional optimization methods cannot solve it. Evolutionary algorithms (EAs) have the potential to address it since they are a kind of population-based heuristic search methods that does not need the gradient information. However, EAs will face the following three issues when solving (12).

1) In (12), we need to optimize the number of UAVs (N), the location of UAV j (X_j and Y_j), and the offloading decision ($a_{i,k}$) and the resource allocation ($f_{i,k}$) for mobile user i. Therefore, $(2(N+M)+1)$ decision variables must be optimized. It is evident that the number of decision variables increases with the increase of M and/or N. Due to the fact that we consider a large number of mobile users in this article, obviously, this is a large-scale optimization problem for EAs [33], [34]. For example, if we consider 1000 mobile users and 100 UAVs, the number of decision variables is 2201.
2) Equation (12) includes an integer decision variable (N), continuous decision variables (X_j, Y_j, and $f_{i,k}$), and binary decision variables ($a_{i,k}$). Thus, it is an optimization problem with mixed decision variables. In the evolutionary computation community, it is a challenging task to solve optimization problems with mixed decision variables [35].
3) The deployment of UAVs and task scheduling are closely coupled. On the one hand, the available execution patterns of a task depend on the deployment of UAVs. This is because a task should be located in the coverage area of a UAV if it is expected to be executed on this UAV. On the other hand, for a given deployment of UAVs, its performance cannot be accurately assessed unless the corresponding task scheduling is optimal.

Therefore, it is inefficient to optimize (12) directly by EAs. In this article, we propose a two-layer optimization method called ToDeTaS, which decomposes (12) into a two-layer optimization problem. To be specific, the upper layer optimizes the deployment of UAVs and the lower layer optimizes the task scheduling. ToDeTaS provides the following technical advantages.

1) In the upper layer, the deployment of UAVs originally involves $(2N+1)$ decision variables. We propose a new encoding mechanism, by which there are only two decision variables in the deployment of UAVs. In addition, in the lower layer, there are originally $2M$ decision variables. For a given deployment of UAVs, the resource allocation in the task scheduling can be obtained through simple derivations. As a result, there are indeed M decision variables in the lower layer. Therefore, the original large-scale optimization problem is decomposed into two optimization problems that can be solved much easier than the original one because they have fewer decision variables.
2) In the upper layer, the optimization problem includes an integer decision variable [N in (12)] and continuous decision variables [X_j and Y_j in (12)]. As analyzed later,

the integer decision variable can be removed by our new encoding mechanism. In addition, the optimization problem in the lower layer includes binary decision variables [$a_{i,k}$ in (12)] and continuous decision variables [$f_{i,k}$ in (12)]. Note that the optimal $f_{i,k}$ in (12) can be easily obtained without any optimization. Therefore, the original optimization problem with mixed decision variables is divided into an optimization problem with continuous decision variables [X_j and Y_j in (12)] in the upper layer and an optimization problem with binary decision variables [$a_{i,k}$ in (12)] in the lower layer. Thus, there do not exist any mixed decision variables in the two-layer optimization problem.

3) In ToDeTaS, we first generate a deployment in the upper layer. Based on the given deployment, it is easy to determine feasible execution patterns of each task; thus, we can obtain the feasible offloading decision with a higher probability. By optimizing task scheduling in the lower layer, we can accurately assess the performance of the deployment of UAVs. Therefore, the upper layer promotes the feasibility of the lower layer, and the lower layer enhances the accuracy of the performance evaluation of the upper layer. As a result, we achieve the joint deployment and task scheduling optimization.

In summary, ToDeTaS is able to address the three aforementioned issues and provides a promising way to use EAs to solve (12).

B. ToDeTaS

When traditional EAs optimize the deployment of UAVs in the upper layer, each individual is usually an entire deployment. As introduced in Section II, UAV j ($j \in \mathcal{N}$) is represented by (X_j, Y_j, H), and X_j and Y_j should be optimized. In addition, the number of UAVs is N. Thus, the length of each individual in traditional EAs is $2N$. Due to the fact that the number of UAVs should be optimized during the evolution, N may change from one generation to another generation. Therefore, in traditional EAs, each individual has a variable length. Under this condition, the deployment of UAVs is a variable-length optimization problem. Currently, it is very challenging for EAs to cope with variable-length optimization problems [36].

We find an interesting phenomenon in the deployment of UAVs: each UAV has two decision variables (i.e., X_j and Y_j ($j \in \mathcal{N}$) in the x-axis and y-axis, respectively), and all elements in $\{X_1, \ldots, X_N\}$ have the same upper and lower bounds, as well as all elements in $\{Y_1, \ldots, Y_N\}$. Based on this observation and inspired by Wang *et al.* [37], we propose a new encoding mechanism: the location of each UAV is encoded into an individual and the entire population denotes an entire deployment, as shown in Fig. 2. This encoding mechanism has the following advantages: 1) each individual has a fixed length during the evolution, rather than a variable length and 2) the length of each individual is equal to two, which means the deployment of UAVs is optimized in a very low-dimensional search space, that is, two.

The general framework of ToDeTaS is presented in Algorithm 1. First, we generate an initial population $\mathcal{P}$ with N individuals (i.e., an initial deployment of UAVs) by Algorithm 2. Afterward, we calculate the offloading decision $\mathbf{a}$ and the resource allocation $\mathbf{f}$ according to $\mathcal{P}$ by Algorithm 5. Subsequently, we evaluate the system energy consumption of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$. During the evolution, if $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is feasible, which means that all tasks can be executed under delay constraints, the elimination operator is implemented in Algorithm 3 to consistently delete one individual until $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is infeasible. Then, we apply DE to produce an offspring population $\mathcal{Q}$. Each individual in $\mathcal{Q}$ is used to update $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ via Algorithm 4. On the one hand, if the updated $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is infeasible, we will check *num_inf*, which denotes the consecutive infeasible number of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$. If *num_inf* reaches a predefined threshold value (i.e., 1000 in this article), which indicates that N cannot be reduced any more, $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ will return to its last feasible status and we will optimize $\{\mathcal{P}, \mathbf{a}, \mathbf{f}\}$ by DE. On the other hand, if the

Algorithm 1 General Framework of ToDeTaS

```
1: $N = N_{max}$; // $N$ denotes the number of UAVs and $N_{max}$ denotes the maximum number of UAVs;
2: Generate an initial population $\mathcal{P}$ with $N$ individuals (i.e., an initial deployment of UAVs) by **Algorithm 2**;
3: Calculate the offloading decision $\mathbf{a}$ and the resource allocation $\mathbf{f}$ according to $\mathcal{P}$ through **Algorithm 5**;
4: Evaluate the system energy consumption of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$;
5: $FEs = 1$; // $FEs$ denotes the number of fitness evaluations
6: $flag = 0$ and $num\_inf = 0$; // $flag$ is the optimization status and $num\_inf$ denotes the consecutive infeasible number of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$
7: while $FEs < FEs_{max}$ do
8:     while $flag = 0$ and $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is feasible do
9:         $\{N_{temp}, \mathcal{P}_{temp}, \mathbf{a}_{temp}, \mathbf{f}_{temp}\} = \{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$;
10:        Perform the elimination operator by **Algorithm 3**;
11:    end while
12:    Implement the mutation and crossover operators of DE to produce an offspring population $\mathcal{Q}$;
13:    for $i = 1, \ldots, N$ do
14:        Utilize the $i$th individual in $\mathcal{Q}$ to update $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ via the updating operator in **Algorithm 4**;
15:        if $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is infeasible then
16:            $num\_inf = num\_inf + 1$;
17:            if $num\_inf = 1000$ then
18:                $flag = 1$;
19:                Return $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ to its last feasible status, i.e., $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\} = \{N_{temp}, \mathcal{P}_{temp}, \mathbf{a}_{temp}, \mathbf{f}_{temp}\}$;
20:                Break;
21:            end if
22:        end if
23:        if $flag = 0$ and $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is feasible then
24:            $num\_inf = 0$;
25:            Break;
26:        end if
27:    end for
28: end while
29: return $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$
```

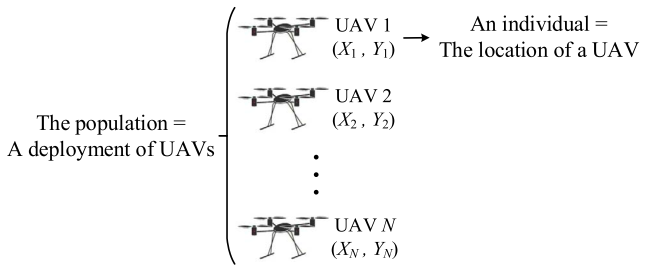


Fig. 2. Encoding mechanism in this article for the deployment of UAVs.

Algorithm 2 Initialization

```
1: $num\_vio = 0$;
2: Generate a location for the first UAV randomly and put it into $\mathcal{P}$;
3: for $j = 2$ to $N$ do
4:     Generate a location for the $j$th UAV randomly;
5:     if the $j$th UAV satisfies $C2$ in (12) then
6:         Put it into $\mathcal{P}$;
7:         $num\_vio = 0$;
8:     else
9:         $num\_vio = num\_vio + 1$;
10:        if $num\_vio > 200$ then
11:            Clear $\mathcal{P}$ and go to Step 1;
12:        end if
13:        Go to Step 4;
14:    end if
15: end for
16: return $\mathcal{P}$
```

updated $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is feasible, we will implement the elimination operator on it. The above process continues until the maximum number of fitness evaluations (FEs_{max}) is met.

In principle, this article achieves the joint deployment and task scheduling optimization through optimizing a 4-tuple: $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$. Moreover, once $num\_inf = 1000$, the optimal value of N is obtained. Under this condition, both steps 8–11 (i.e., the elimination operator) and steps 15–26 are unnecessary, and we only concentrate on the optimization of $\{\mathcal{P}, \mathbf{a}, \mathbf{f}\}$ by DE (i.e., steps 12–14 and 27).

It is noteworthy that the deployment of UAVs in the upper layer depends on Algorithms 3 and 4, and the task scheduling in the lower layer depends on Algorithm 5.

C. Initialization

Algorithm 2 introduces the initialization of $\mathcal{P}$, which contains the locations of N UAVs. First, we randomly generate a location for the first UAV and put it into $\mathcal{P}$. After that, we generate a location for the second UAV. If this UAV satisfies $C2$ in (12), which suggests that the distance between the first and second UAVs is not smaller than d_{min}^{UU} and they will not collide, then we put it into $\mathcal{P}$. Otherwise, the generation of the location of the second UAV is unsuccessful. Under this condition, if the consecutive unsuccessful number is bigger than 200, we restart the initialization; otherwise, the location of the second UAV is regenerated by step 4. Subsequently, we execute the above process on the third UAV and so forth. Finally, all UAVs' locations are successfully generated and an initial deployment of UAVs is obtained (i.e., $\mathcal{P}$).

D. Upper-Layer Optimization

The aim of the upper-layer optimization is to determine the optimal deployment of UAVs, in other words, the optimal number and locations of UAVs. In ToDeTaS, the number of UAVs is equal to the population size of $\mathcal{P}$ (i.e., N). Therefore, the optimization of the number of UAVs is equivalent to the adjustment of N. By analyzing the multi-UAV-enabled MEC system proposed in this article, the following property is obtained.

Property 1: The number of UAVs should be as small as possible under the condition that all tasks can be executed under delay constraints.

Algorithm 3 Elimination Operator

```
1: Choose two individuals with the minimum Euclidean distance from $\mathcal{P}$,
   then calculate their second minimum Euclidean distances and delete the
   one with smaller second minimum Euclidean distance from $\mathcal{P}$. If they
   have the same second minimum Euclidean distance, then we calculate
   their third minimum Euclidean distances and so forth;
2: $N = N - 1$;
3: Calculate $\mathbf{a}$ and $\mathbf{f}$ according to $\mathcal{P}$ based on Algorithm 5;
4: Evaluate the system energy consumption of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$;
5: $FEs = FEs + 1$;
6: return $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ and $FEs$
```

Algorithm 4 Updating Operator

```
1: Utilize the $i$th individual in $\mathcal{Q}$ to replace a randomly selected individual
   in $\mathcal{P}$ and obtain a new population $\mathcal{R}$;
2: if $\mathcal{R}$ satisfies $C2$ in (12) then
3:     Calculate the offloading decision $\mathbf{a}'$ and the resource allocation $\mathbf{f}'$
       according to $\mathcal{R}$ based on Algorithm 5;
4:     Evaluate the system energy consumption of $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$;
5:     $FEs = FEs + 1$;
6:     Denote the numbers of completed tasks of $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$ and
       $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ as $NC\_R$ and $NC\_P$, respectively, and denote the energy
       consumption of $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$ and $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ as $EC\_R$ and $EC\_P$,
       respectively;
7:     if $NC\_R > NC\_P$ then
8:         $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\} = \{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$;
9:     else if $NC\_R == NC\_P == M$ and $EC\_R < EC\_P$ then
10:        $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\} = \{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$;
11:    end if
12: end if
13: return $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ and $FEs$
```

We analyze the rationality of this property in the Appendix. Based on this property, the population size of $\mathcal{P}$ is adjusted as follows: we first set the initial N as the maximum number of UAVs ($N_{max} = M/n_{max}$), and then N is gradually decreased by the elimination operator in Algorithm 3 until at least one task cannot be executed under delay constraints. As shown in Algorithm 3, in each time, we only delete one individual from $\mathcal{P}$. A question is which individual should be deleted. In this article, we consider that the most crowded individual should be deleted. It is because the UAV corresponding to this individual may be redundant, thus adding system energy consumption.

The second issue in the upper-layer optimization is to determine the optimal locations of UAVs, which is achieved by making use of DE. In this article, a classical DE version, DE/rand/1/bin [38], is adopted. For the ith individual $\vec{x}_i = (x_{i,1}, x_{i,2})$ $(i \in \{1, \ldots, N\})$ in $\mathcal{P}$, the mutation and crossover operators of DE/rand/1/bin are introduced as follows:

$$\vec{v}_i = \vec{x}_{r1} + F * (\vec{x}_{r2} - \vec{x}_{r3}) \tag{13}$$

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand_j(0,1) \leq CR \quad \text{or} \quad j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{14}$$

where $i \in \{1, \ldots, N\}$; $j \in \{1, 2\}$; $\vec{x}_{r1}$, $\vec{x}_{r2}$, and $\vec{x}_{r3}$ are the three mutually distinct individuals randomly selected from $\mathcal{P}$; $\vec{v}_i = (v_{i,1}, v_{i,2})$ and $\vec{u}_i = (u_{i,1}, u_{i,2})$ are the mutant vector and the trial vector, respectively; $u_{i,j}$, $v_{i,j}$, and $x_{i,j}$ are the jth dimension of $\vec{u}_i$, $\vec{v}_i$, and $\vec{x}_i$, respectively; F is the scaling factor; j_{rand} is an integer randomly selected between 1 and 2 to ensure that $\vec{u}_i$ is different from $\vec{x}_i$ in at least one dimension; $rand_j(0, 1)$ denotes a uniformly distributed random number between 0 and 1 for each j; and CR is the crossover control parameter.

Algorithm 5 Task Scheduling

```
1: Calculate $\mathbf{f}$ based on the given $\mathcal{P}$;
2: Divide the tasks into three categories. Suppose that the first, second, and
   third categories have $M_1$, $M_2$, and $M_3$ tasks, respectively;
3: Initialize $\mathbf{a} = \mathbf{0}$ in (20);
4: For the tasks in the first category, $a_{1,0} = \cdots = a_{M_1,0} = 1$;
5: $A = \{1, \ldots, M_2\}$;
6: while $A \neq \emptyset$ do
7:     Choose the task with the minimum number of candidate patterns in
       the second category (denoted as the $s$th task);
8:     Suppose that this task has $n_s$ candidate patterns and the corresponding
       minimum energy consumption of these $n_s$ candidate patterns is:
       $E^{\star}_{s,1}, \ldots, E^{\star}_{s,n_s}$;
9:     The candidate pattern with $min(E^{\star}_{s,1}, \ldots, E^{\star}_{s,n_s})$ is selected, denoted
       as $c_s$.
10:    $a_{M_1+s,c_s} = 1$ in $\mathbf{a}$ and $A = A \backslash \{s\}$;
11:    The number of tasks that the $c_s$th UAV can serve is reduced by one,
       and the candidate pattern sets of the rest of the tasks in $A$ are updated;
12:    if the candidate pattern sets of all the tasks in $A$ are empty then
13:        Break;
14:    end if
15: end while
16: $B = \{1, \ldots, M_3\}$;
17: while $B \neq \emptyset$ do
18:    Suppose that $U_i$ $(i = 1, \ldots, |B|)$ in the third category has $n_i$ candidate
       patterns, and the corresponding minimum energy consumption of these
       $n_i$ candidate patterns is: $E^{\star}_{i,1}, \ldots, E^{\star}_{i,n_i}$;
19:    Normalize $n_i$ and $(E^{\star}_{i,1}, \ldots, E^{\star}_{i,n_i})$ of $U_i$ $(i = 1, \ldots, |B|)$: $nor(n_i)$ and
       $(nor(E^{\star}_{i,1}), \ldots, nor(E^{\star}_{i,n_i}))$;
20:    Compute $nor(n_i) * nor(E^{\star}_{i,1}), \ldots, nor(n_i) * nor(E^{\star}_{i,n_i})$ for $U_i$ $(i =
       1, \ldots, |B|)$. Thus, we can obtain $\sum_{i=1}^{|B|} n_i$ values. By selecting the mini-
       mum value, we can determine the corresponding task (denoted as the
       $s$th task in the third category) and pattern (denoted as $c_s$);
21:    $a_{M_1+M_2+s,c_s} = 1$ in $\mathbf{a}$ and $B = B \backslash \{s\}$;
22:    if $U_s$ is executed on a UAV then
23:        The number of tasks that this UAV can serve is reduced by one, and
           the candidate pattern sets of the rest of the tasks in $B$ are updated;
24:    end if
25: end while
26: return $\{\mathbf{a}, \mathbf{f}\}$
```

During the evolution, DE is implemented on $\mathcal{P}$ to produce an offspring population $\mathcal{Q}$. Thereafter, each individual in $\mathcal{Q}$ is utilized to replace a randomly selected individual in $\mathcal{P}$; thus, $\mathcal{P}$ is updated, denoted as $\mathcal{R}$. For $\mathcal{R}$, if it satisfies $C2$ in (12), we compute the offloading decision $\mathbf{a}'$ and the resource allocation $\mathbf{f}'$. If $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$ can execute more tasks under delay constraints than $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$, or if both of them can execute all tasks under delay constraints and the system energy consumption of $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$ is less than that of $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$, then $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is replaced with $\{N, \mathcal{R}, \mathbf{a}', \mathbf{f}'\}$. The updating operator is given in Algorithm 4.

Regarding steps 15–22 and 23–26 in Algorithm 1, we would like to give the following remarks.

1) *Steps 15–22: flag* represents the optimization status. Specifically, *flag* $= 0$ denotes that the elimination operator can be implemented; instead, *flag* $= 1$ denotes that the elimination operator will not be used any more. If $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is still infeasible after 1000 consecutive updates, we consider that N cannot be reduced and the optimal number of UAVs has been found (i.e., $N+1$). Thus, we let *flag* $= 1$. Under this condition, the followings steps will be applied: $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ returns to its last feasible status, the elimination operator is no longer used, and we continue to take advantage of the updating operator to optimize $\mathcal{P}$, that is, the locations of UAVs.
2) *Steps 23–26:* If *flag* $= 0$ and $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ is feasible, the updating operator breaks and the elimination operator is implemented on $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ to further reduce N.

Overall, in the upper-layer optimization, the number of UAVs is optimized by the elimination operator and the locations of UAVs are optimized by the updating operator. Moreover, steps 15–22 and 23–26 control the switch between the elimination operator and the updating operator.

E. Lower-Layer Optimization

The lower-layer optimization aims to optimize the task scheduling under a given deployment of UAVs, including the offloading decision and the resource allocation. For a given deployment of UAVs, N, X_j, Y_j $(j \in \mathcal{N})$, and E^H are fixed in (12). In addition, this deployment must satisfy $C2$ in (12) since if it does not satisfy $C2$, it cannot enter the population as shown in step 2 of Algorithm 4. Therefore, we only need to focus on $a_{i,k}$ and $f_{i,k}$ $(i \in \mathcal{M}$ and $k \in \mathcal{K})$ in (12). By substituting (2) and (10), which are related to $f_{i,k}$, to (12), the lower-layer optimization problem can be expressed as

$$\min_{a_{i,k}, f_{i,k}} \sum_{i=1}^{M} \left(a_{i,0}\eta_1 (f_{i,0})^{\nu-1} C_i + \sum_{k=1}^{N} a_{i,k} \left(P\frac{D_i}{r_{i,k}} + \eta_2 (f_{i,k})^{\nu-1} C_i \right) \right)$$
$$\text{s.t. } C1,\ C3,\ C4,\ C5,\ C6,\ C7,\ \text{and } C8. \tag{15}$$

It can be seen from (15) that the more the computation resources consumed to complete a task under a certain pattern (i.e., $f_{i,k}$), the greater the energy consumption [the objective function in (15)]. It is because the energy consumption increases monotonously with the increase of $f_{i,k}$. Therefore, to minimize the energy consumption, $f_{i,k}$ should be as small as possible. However, when U_i is executed in pattern k, to ensure that delay constraints $C7$ and $C8$ are satisfied, $f_{i,k}$ must not be smaller than a minimum value, which can be calculated based on $C7$ and $C8$.

Substituting (1) and (9) to $C7$ and $C8$, respectively, one can obtain that:

1) when U_i is executed in pattern 0
$$f_{i,0} \geq \frac{C_i}{T}, \quad \forall i \in \mathcal{M} \tag{16}$$
2) when U_i is executed in pattern k
$$f_{i,k} \geq \frac{C_i}{T - \frac{D_i}{r_{i,k}}}, \quad \forall i \in \mathcal{M}, k \in \mathcal{K}\backslash\{0\}. \tag{17}$$

From (16) and (17), the minimum computation resources are (C_i/T) and $[C_i/(T-(D_i/r_{i,k}))]$, respectively, when U_i is executed in pattern 0 and the other patterns. Thus, each element of the optimal resource allocation $\mathbf{f}$ can be given as

$$f^{\star}_{i,k} = \begin{cases} \frac{C_i}{T}, & \text{if } a_{i,k} = 1,\ k = 0 \\ \frac{C_i}{T - \frac{D_i}{r_{i,k}}}, & \text{if } a_{i,k} = 1,\ k > 0 \\ 0, & \text{otherwise} \end{cases}, \quad \forall i \in \mathcal{M},\ k \in \mathcal{K}. \tag{18}$$

After obtaining the optimal resource allocation, $C5$, $C6$, $C7$, and $C8$ are satisfied, and then we can rewrite the lower-layer optimization problem again by substituting (18) to (15)

$$\begin{aligned}\min_{a_{i,k}} \ & \sum_{i=1}^{M}\Bigg(a_{i,0}\eta_1\big(f_{i,0}^{\star}\big)^{v-1}C_i \\ & \quad + \sum_{k=1}^{N} a_{i,k}\left(P\frac{D_i}{r_{i,k}} + \eta_2\big(f_{i,k}^{\star}\big)^{v-1}C_i\right)\Bigg) \\ \text{s.t. } & C1,\ C3,\ \text{and } C4. \end{aligned} \tag{19}$$

Remark 1: As mentioned in the Appendix, $E_{i,k}^{\star}$ represents the minimum energy to complete U_i $(i \in \mathcal{M})$ in pattern k $(k \in \mathcal{K})$. Actually, $E_{i,0}^{\star} = \eta_1(f_{i,0}^{\star})^{v-1}C_i$, and $E_{i,k}^{\star} = P(D_i/r_{i,k}) + \eta_2(f_{i,k}^{\star})^{v-1}C_i$, $k \in \mathcal{K}\backslash\{0\}$.

As can be seen, we only need to optimize $a_{i,k}$; thus, (19) is a 0-1 integer programming problem since $a_{i,k} = 0$ or 1. Although classical mathematical programming methods, such as the branch-and-bound algorithm [39], can be used to solve (19), they are time-consuming due to the large-scale characteristic in this article. To this end, we propose a greedy algorithm to efficiently obtain the near-optimal solution of (19).

First, we define a candidate pattern set for each task.

1) This task can be executed in each candidate pattern under delay constraints.
2) If pattern 0 is one of the candidate patterns, then the energy consumption of any other candidate pattern is less than that of pattern 0.

Subsequently, all tasks are divided into three categories.

1) The tasks' candidate pattern sets only contain pattern 0.
2) The tasks' candidate pattern sets do not contain pattern 0.
3) The tasks' candidate pattern sets contain both pattern 0 and other patterns.

Suppose that there are M_1, M_2, and M_3 tasks in the first, second, and third categories, respectively. Then, offloading decision $\mathbf{a}$ is expressed in the following, each element of which is initialized to be zero:

$$\mathbf{a} = \begin{pmatrix} a_{1,0} & a_{1,1} & \cdots & a_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M_1,0} & a_{M_1,1} & \cdots & a_{M_1,N} \\ a_{M_1+1,0} & a_{M_1+1,1} & \cdots & a_{M_1+1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M_1+M_2,0} & a_{M_1+M_2,1} & \cdots & a_{M_1+M_2,N} \\ a_{M_1+M_2+1,0} & a_{M_1+M_2+1,1} & \cdots & a_{M_1+M_2+1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M_1+M_2+M_3,0} & a_{M_1+M_2+M_3,1} & \cdots & a_{M_1+M_2+M_3,N} \end{pmatrix}. \tag{20}$$

We give the priorities of these three categories in the descending order. The tasks in the first category have the highest priority. This is because they are executed locally (i.e., $a_{1,0} = \cdots = a_{M_1,0} = 1$) and do not consume any computation resources from MEC servers on UAVs. Since the tasks in the second category can only be executed on UAVs, we need to give them the second highest priority, with the aim of completing as many tasks as possible. In addition, the tasks in the third category can be executed locally in the worst case. Therefore, they have the lowest priority.

Next, we determine the offloading decision for the tasks in the second and third categories.

1) The offloading decision for the tasks in the second category is given in steps 5–15 in Algorithm 5. When determining which task to execute, we first select the task with the minimum number of candidate patterns, the aim of which is to complete all tasks with a higher probability. Afterward, we choose one of the candidate patterns of this task by considering their minimum energy consumption.
2) The offloading decision for the tasks in the third category is given in steps 16–25 in Algorithm 5. When determining which task to execute, we consider the number of candidate patterns and the energy consumption simultaneously. We prefer the tasks with fewer candidate patterns and less energy consumption; thus, all tasks can be completed with the system energy consumption being as little as possible.

Remark 2: Only in the second category, some tasks may not be executed under delay constraints. For the other two categories, all tasks can be definitely completed.

Remark 3: In Algorithm 4, it is necessary to compute the number of completed tasks. Note that the number of uncompleted tasks is equal to the number of the remaining tasks in A when Algorithm 5 terminates, that is, the number of rows in $\mathbf{a}$, in which all the elements are zero.

F. Discussion

The proposed ToDeTaS has the following characteristics.

1) This article optimizes a 4-tuple: $\{N, \mathcal{P}, \mathbf{a}, \mathbf{f}\}$ to minimize the energy consumption of the proposed multi-UAV-enabled MEC system.
2) By mining the specific-knowledge of this system, we propose a new encoding mechanism and adaptively adjust population size N (i.e., the number of UAVs).
3) DE serves as the search engine to optimize $\mathcal{P}$, that is, the locations of UAVs.
4) By exploiting the correlation between the upper layer and the lower layer, for a given deployment of UAVs in the upper layer, we directly derive the optimal $\mathbf{f}$ and propose a greedy algorithm to efficiently optimize $\mathbf{a}$.
5) ToDeTaS includes few parameters: the scaling factor F and the crossover control parameter CR in DE. Moreover, due to the low-dimensional search space, F and CR are not sensitive.

The novelties of this article can be summarized as follows.

1) This article is the first attempt to establish a multi-UAV-enabled MEC system to serve large-scale mobile users.
2) An optimization problem is formulated to jointly optimize the deployment of UAVs and the task scheduling. The main challenges of this optimization problem are two-fold: a) large-scale mixed decision variables and

TABLE I
PARAMETER SETTINGS IN THE MULTI-UAV-ENABLED MEC SYSTEM PROPOSED IN THIS ARTICLE

Parameter	Value	Parameter	Value
$C_i,\ i \in \mathcal{M}$	[16, 1600]MCycles	d_{min}^{UU}	10m
$D_i,\ i \in \mathcal{M}$	[10, 1000]KB	n_{max}	10
H	100m	B	1MHz
θ	$\frac{\pi}{4}$	P	1W
$f_{i,0},\ i \in \mathcal{M}$	[0, 0.8]GHz	β_0	1.42×10^{-4}
$f_{i,k},\ i \in \mathcal{M},\ k \in \mathcal{K}\backslash\{0\}$	[0, 10]GHz	G_0	2.2846
η_1	10^{-27}	N_0	10^{-20}W/Hz
η_2	10^{-28}	P_0	1000W
v	3	T	1s

TABLE II
SIDE LENGTHS OF SQUARE AREAS WITH DIFFERENT NUMBERS OF MOBILE USERS

M	100	200	300	400	500
Side Length (m)	320	450	550	640	710
M	600	700	800	900	1000
Side Length (m)	780	840	900	950	1000

TABLE III
EXPERIMENTAL RESULTS OF DE-VND AND TODE-VND IN TERMS OF MEAN NC AND SR

M	DE-VND		ToDE-VND	
	Mean NC (Std Dev)	SR	Mean NC (Std Dev)	SR
100	43.07 (1.39)	0.00%	100.00 (0.00)	100.00%
200	81.07 (3.41)	0.00%	200.00 (0.00)	100.00%
300	105.37 (8.86)	0.00%	300.00 (0.00)	100.00%
400	129.13 (12.53)	0.00%	399.97 (0.18)	96.67%
500	145.20 (19.00)	0.00%	499.63 (0.48)	63.33%
600	163.23 (22.38)	0.00%	598.67 (0.75)	6.67%
700	182.90 (31.13)	0.00%	698.47 (0.88)	13.33%
800	199.87 (37.86)	0.00%	797.33 (1.01)	0.00%
900	228.13 (44.69)	0.00%	897.03 (0.84)	0.00%
1000	247.87 (53.49)	0.00%	997.17 (1.07)	0.00%

b) strong coupling between the deployment of UAVs and the task scheduling.

3) We propose a new two-layer optimization method called ToDeTaS. The two-layer structure is able to deal with large-scale decision variables and strong coupling between the deployment of UAVs and the task scheduling. Moreover, in the upper layer, an integer decision variable (i.e., the number of UAVs) and continuous decision variables (i.e., the locations of UAVs) are handled by a new encoding mechanism and an elimination operator, and in the lower layer, binary decision variables (i.e., offloading decision of each mobile user) and continuous decision variables (i.e., resource allocation of each mobile user) are tackled by an efficient greedy algorithm. Therefore, ToDeTaS can also address the challenge caused by the mixed decision variables.

IV. EXPERIMENTAL STUDY

A. Experimental Settings

The parameter settings of the proposed multi-UAV-enabled MEC system are given in Table I [30], [40], [41]. In addition, we applied ten instances with different numbers of mobile users to study the performance of ToDeTaS: $M = 100, 200, \ldots, 1000$. We assumed that all mobile users were distributed in square areas with different side lengths, as shown in Table II.

The proposed ToDeTaS includes two parameters, which were set as follows: $F = 0.9$ and $CR = 0.9$. The maximum number of fitness evaluations (FEs_{max}) was set to 10 000 and 30 independent runs were implemented on ToDeTaS.

In order to compare the performance of different algorithms, three performance indicators were employed.

1) The first performance indicator was the average number of completed tasks and the standard deviation over 30 independent runs (denoted as "Mean NC" and "Std Dev").
2) The second performance indicator was the success rate (denoted as SR), which means the percentage of successful runs over 30 independent runs. A run is successful if all tasks can be completed when an algorithm ends.
3) If SR of an algorithm is equal to 100%, then we compute the system energy consumption via (12) (denoted as EC). Thus, the third performance indicator was the average system energy consumption (i.e., $(1/30)\sum_{i=1}^{30} \text{EC}_i$, where EC_i represents the system energy consumption of the ith independent run) and the standard deviation over 30 independent runs (denoted as Mean EC and Std Dev).

B. Effectiveness of Two-Layer Optimization

We handle the joint deployment and task scheduling optimization by a two-layer optimization. Moreover, we optimize the deployment of UAVs and task scheduling in the upper and lower layers, respectively. To verify the effectiveness of the two-layer optimization, we solved (12) by a single-layer optimization method proposed in [42]. The method in [42] is designed to solve optimization problems with a variable number of dimensions, in which different individuals in the population have different lengths, and the length of each individual is updated according to a probability-based way. As pointed out in Section III-B, the deployment of UAVs is a variable-length optimization problem due to the fact that the optimal number of UAVs is unknown. Therefore, the method in [42] was chosen to solve (12) as a single-layer optimization method in this article. Note that we made a simple revision to this method by replacing particle swarm optimization with DE as the search engine. The resultant method was called DE-VND. In DE-VND, each initial individual included a deployment of UAVs as well as task scheduling, both of them were randomly generated. In addition, we designed a two-layer version of DE-VND, called ToDE-VND. In ToDE-VND, the deployment of UAVs in the upper layer was the same with DE-VND; however, the task scheduling in the lower layer was the same with ToDeTaS.

The performance of DE-VND was compared with that of ToDE-VND on the ten instances. In the experiments, DE-VND and ToDE-VND had the same parameter settings: the population size was set to 100; the probabilities p_1 to $\vec{x}_i$, p_2 to $\vec{x}_{r1}$, p_3 to $\vec{x}_{r2}$, and p_4 to $\vec{x}_{r3}$ were set to 0.25, 0.25, 0.25 and 0.25, respectively; $F = 0.9$, $CR = 0.9$, $FEs_{max} = 10\,000$, and 30 independent runs were implemented. The experimental results in terms of mean NC and SR are summarized in Table III.

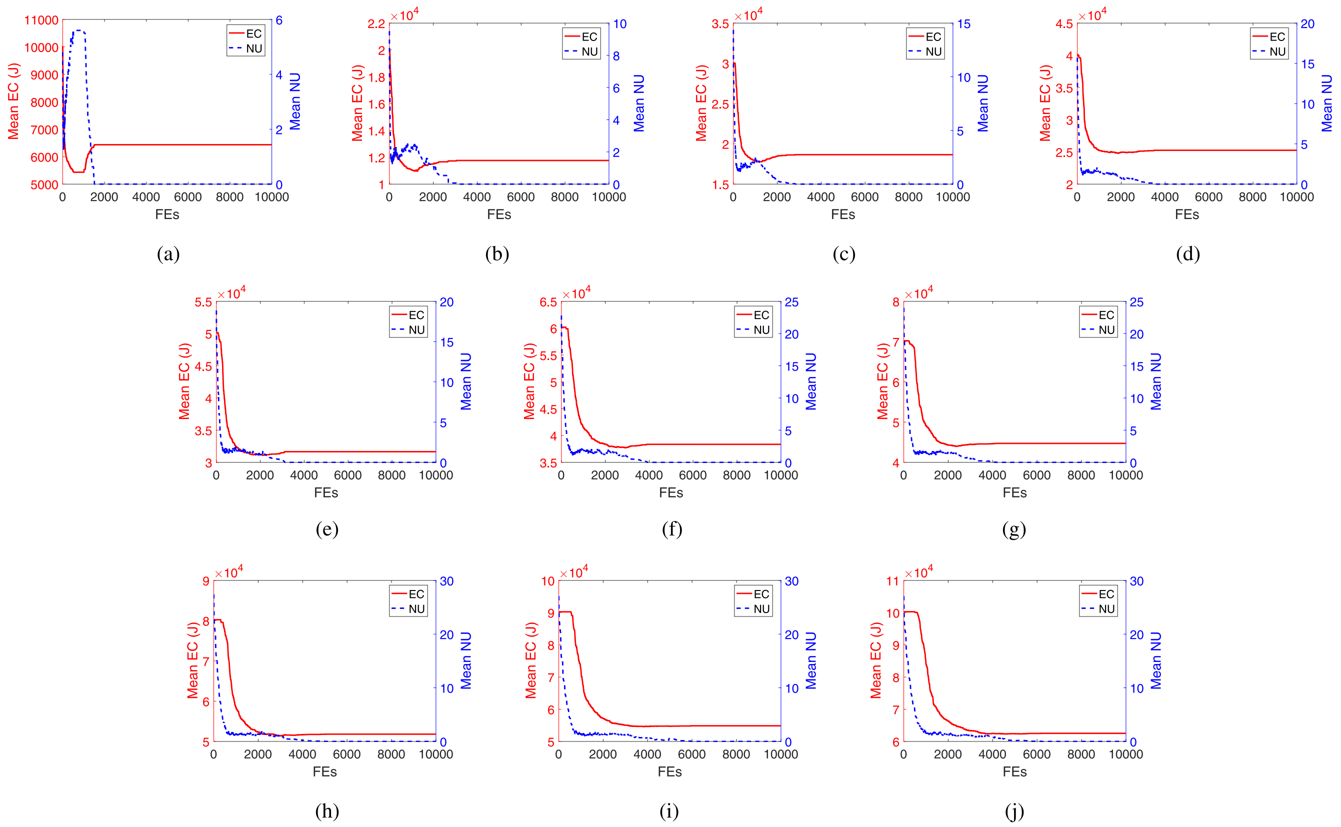


Fig. 3. Evolution of the mean EC (J) and the mean NU provided by ToDeTaS on the ten instances. (a) $M = 100$. (b) $M = 200$. (c) $M = 300$. (d) $M = 400$. (e) $M = 500$. (f) $M = 600$. (g) $M = 700$. (h) $M = 800$. (i) $M = 900$. (j) $M = 1000$.

From Table III, as far as the mean NC is concerned, ToDE-VND is significantly better than DE-VND on all instances. In addition, ToDE-VDE provides higher SR than DE-VND from $M = 100$ to 700. Moreover, when $M = 100$, 200, and 300, ToDE-VND achieves 100% SR. The superiority of ToDE-VND against DE-VND can be attributed to two aspects: 1) by the two-layer optimization, (12) is decomposed into two optimization problems in the upper and lower layers with fewer numbers of decision variables and 2) in ToDE-VND, the lower layer is generated based on the upper layer, which in turn enhances the accuracy of the evaluation of the upper layer; thus, the correlation between the upper and lower layers has been considered. However, in DE-VND, the deployment of UAVs and task scheduling are optimized independently. In this case, on the one hand, for a given deployment of UAVs, the probability that all tasks can be completed in the task scheduling remarkably decreases. On the other hand, we cannot provide an accurate evaluation of the deployment of UAVs based on the corresponding task scheduling. The aforementioned discussion verifies the effectiveness of the two-layer optimization, which is the main motivation of this article.

C. Effectiveness of Upper-Layer Optimization

The difference between ToDE-VND and ToDeTaS is the upper-layer optimization. To be specific, ToDE-VND and ToDeTaS have different encoding mechanisms and different ways to deal with the variable-length optimization problem in (12). Hence, by comparing ToDE-VND with ToDeTaS, we can study the effectiveness of the upper-layer optimization. It can be seen from Section IV-B that ToDE-VND and ToDeTaS have the same parameter settings.

Table IV reports the experimental results derived from ToDE-VND and ToDeTaS in terms of mean NC, SR, and mean EC. When $M = 100$, 200, and 300, both ToDE-VND and ToDeTaS can complete all tasks and provide 100% SR. Under this condition, we compared their mean EC. It is clear that the mean EC values resulting from ToDeTaS are significantly smaller than those of ToDE-VND. In addition, for the remaining instances, ToDeTaS succeeds in completing all tasks consistently. In contrast, ToDE-VND's SR is smaller than 100% on each instance. More importantly, ToDE-VND fails to complete all tasks in each run for the instances with a larger number of mobile users, that is, $M = 800$, 900, and 1000. The reason why ToDeTaS performs better than ToDE-VND is straightforward: the former searches for the optimal deployment of UAVs in the search space with a much lower dimension compared with the latter. Moreover, ToDeTaS encodes the location of a UAV into an individual, thus transforming a variable-length optimization problem into a fixed-length one. It is noteworthy that ToDeTaS adopts an elimination operator to adaptively adjust the population size. As a result, an important parameter, that is, the population size, has been eliminated.

Fig. 3 plots the evolution of the mean EC and the mean number of uncompleted tasks (denoted as mean NU) provided

TABLE IV
EXPERIMENTAL RESULTS OF TODE-VND AND TODETAS IN TERMS OF MEAN NC, SR, AND MEAN EC (J)

M	ToDE-VND			ToDeTaS		
	Mean NC (Std Dev)	SR	Mean EC (Std Dev)	Mean NC (Std Dev)	SR	Mean EC (Std Dev)
100	100.00 (0.00)	100.00%	7568.24 (498.52)	100.00 (0.00)	100.00%	6435.16 (489.34)
200	200.00 (0.00)	100.00%	17361.06 (640.21)	200.00 (0.00)	100.00%	11761.12 (861.58)
300	300.00 (0.00)	100.00%	27821.53 (1030.27)	300.00 (0.00)	100.00%	18654.89 (1144.66)
400	399.97 (0.18)	96.67%	/	400.00 (0.00)	100.00%	25252.44 (1498.72)
500	499.63 (0.48)	63.33%	/	500.00 (0.00)	100.00%	31657.92 (1585.73)
600	598.67 (0.75)	6.67%	/	600.00 (0.00)	100.00%	38360.55 (1713.31)
700	698.47 (0.88)	13.33%	/	700.00 (0.00)	100.00%	44710.19 (1687.06)
800	797.33 (1.01)	0.00%	/	800.00 (0.00)	100.00%	51811.82 (2523.99)
900	897.03 (0.84)	0.00%	/	900.00 (0.00)	100.00%	54858.37 (1817.12)
1000	997.17 (1.07)	0.00%	/	1000.00 (0.00)	100.00%	62516.68 (2471.07)

TABLE V
EXPERIMENTAL RESULTS OF TODETAS-BB AND TODETAS IN TERMS OF MEAN EC (J) AND MEAN RUNTIME (S)

M	Mean EC (Std Dev)		Mean Runtime	
	ToDeTaS-BB	ToDeTaS	ToDeTaS-BB	ToDeTaS
100	6536.17 (499.98)	6435.16 (489.34)	93.60	4.56
200	11763.66 (862.13)	11761.12 (861.58)	141.87	9.52
300	18326.72 (1054.57)	18654.89 (1144.66)	250.86	21.05
400	24093.02 (1224.23)	25252.44 (1498.72)	435.97	42.42
500	30298.00 (1383.99)	31657.92 (1585.73)	726.57	67.30
600	36501.72 (1715.53)	38360.55 (1713.31)	1166.16	106.31
700	41486.40 (1610.91)	44710.19 (1687.06)	1741.31	150.84
800	48690.46 (2906.08)	51811.82 (2523.99)	2603.06	213.90
900	52204.68 (1913.62)	54858.37 (1817.12)	3562.85	251.45
1000	58499.03 (2587.03)	62516.68 (2471.07)	4915.91	329.12
/	/	/	MAR	13.22

by ToDeTaS over 30 independent runs on the ten instances. As shown in Fig. 3, ToDeTaS can consistently complete all tasks and converge after 5000 fitness evaluations.

D. Effectiveness of Lower-Layer Optimization

The lower-layer optimization involves the offloading decision and resource allocation. Although in ToDeTaS, the resource allocation can be determined by simple mathematical derivations, the offloading decision is still a large-scale 0-1 integer programming problem due to a large number of mobile users in this article. To reduce the computational time complexity, we propose a greedy algorithm to solve this problem. One may be interested in the performance difference between our greedy algorithm and other classical mathematical programming methods. To this end, we designed a variant of ToDeTaS, called ToDeTaS-BB, in which the offloading decision was solved by the branch-and-bound algorithm [39]. We implemented the branch-and-bound algorithm via the MATLAB toolbox.

The experimental results of ToDeTaS and ToDeTaS-BB are presented in Table V in terms of mean EC and mean runtime. We can observe from Table V that from $M = 300$ to 1000, overall, ToDeTaS-BB provides slightly less mean EC than ToDeTaS. It is largely because the branch-and-bound algorithm can generate a better offloading decision than the greedy algorithm and improve the accuracy of evaluation for the upper layer. It is interesting to note that ToDeTaS is better than ToDeTaS-BB in terms of the mean EC for a small number of mobile users, that is, $M = 100$ and 200. This phenomenon is not difficult to understand since for a small number of mobile users, the greedy algorithm is able to generate a high-quality offloading decision. In addition, the branch-and-bound algorithm cannot guarantee the absolute optimal offloading decision in the MATLAB toolbox.

With respect to the mean runtime, it is obvious that ToDeTaS performs much faster than ToDeTaS-BB. In this article, we defined the mean accelerator rate (MAR) of ToDeTaS against ToDeTaS-BB

$$\text{MAR} = \frac{1}{10}\sum_{i=1}^{10}\frac{T1_i}{T2_i} \tag{21}$$

where $T1_i$ and $T2_i$ represent the runtime of ToDeTaS-BB and ToDeTaS on the ith instance, respectively. As shown in Table V, ToDeTaS is on average 13.22 times more efficient than its competitor. After a task is executed, the computational time complexity of the greedy algorithm in Algorithm 5 depends mainly on updating the candidate pattern sets of the remaining tasks in steps 11 and 23, which requires MN judgements in the worst case. Due to the fact that there are M tasks, the computational time complexity of the greedy algorithm is $O(M^2N)$ in the worst case. In contrast, the computational time complexity of the branch-and-bound algorithm is $O((N+1)^M)$. Therefore, we can conclude that the greedy algorithm can efficiently optimize the offloading decision with only a slight sacrifice of the system energy consumption, compared with the branch-and-bound algorithm.

E. Effectiveness of Our Multi-UAV-Enabled MEC System

Finally, we compared two algorithms—ToDeTaS-L and ToDeTaS-M—with ToDeTaS to verify the effectiveness of our multi-UAV-enabled MEC system. For ToDeTaS-L and ToDeTaS-M, all tasks can only be executed locally and on UAVs, respectively. However, for ToDeTaS, a task can be executed locally or on a UAV.

Table VI shows the experimental results of ToDeTaS-L, ToDeTaS-M, and ToDeTaS in terms of mean NC over 30 independent runs. As depicted in Table VI, both ToDeTaS-L and ToDeTaS-M cannot successfully complete all tasks on any instance. However, ToDeTaS has the capability to complete all tasks on all ten instances. The poor performance of ToDeTaS-L and ToDeTaS-M can be explained as follows. For ToDeTaS-L, if a mobile device cannot complete its task due to the lack of enough computational resources, then ToDeTaS-L

TABLE VI
EXPERIMENTAL RESULTS OF ToDeTaS-L, ToDeTaS-M, AND ToDeTaS IN TERMS OF MEAN NC

M	Mean NC		
	ToDeTaS-L	ToDeTaS-M	ToDeTaS
100	42.00	99.83	100.00
200	103.00	199.20	200.00
300	149.00	298.63	300.00
400	200.00	397.53	400.00
500	244.00	496.57	500.00
600	283.00	595.67	600.00
700	334.00	693.77	700.00
800	378.00	792.20	800.00
900	455.00	891.23	900.00
1000	515.00	989.00	1000.00

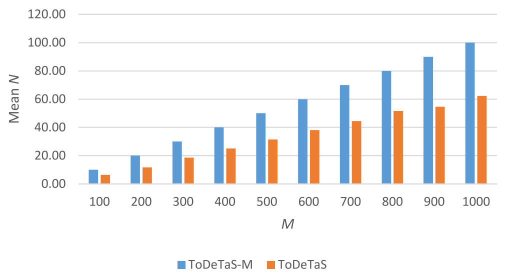


Fig. 4. Experimental results of ToDeTaS-M and ToDeTaS in terms of mean N.

will fail. In addition, for ToDeTaS-M, due to the nonuniform distribution of mobile users, some tasks may not be covered by any UAV; thus, ToDeTaS-M may fail. In principle, ToDeTaS can alleviate the limitations of these two algorithms.

Note that the mean NC values provided by ToDeTaS-M are close to those of ToDeTaS in Table VI. In order to further identify the performance difference, we compared the mean number of UAVs (i.e., mean N) of ToDeTaS-M and ToDeTaS on the ten instances. Fig. 4 shows the experimental results. From Fig. 4, ToDeTaS is able to complete all tasks while requiring considerably fewer UAVs than ToDeTaS-M on each instance. This is because about 40% of the tasks can be executed locally under delay constraints according to the parameter settings in Table I. Therefore, ToDeTaS is capable of reducing about 40% of UAVs compared with ToDeTaS-M. This comparison confirms the effectiveness of our multi-UAV-enabled MEC system.

V. DISCUSSION

A. On Hyper-Heuristic Approaches for the Task Scheduling in the Lower Layer

One may be interested in whether hyper-heuristic approaches (e.g., genetic programming and particle swarm optimization) can work better for the task scheduling in the lower layer. There is no doubt that hyper-heuristic approaches are able to solve the task scheduling in the lower layer and may even obtain a better solution than our greedy algorithm. However, the computational time complexity of a hyper-heuristic approach is significantly higher than that of our greedy algorithm. This is because a hyper-heuristic approach searches for the optimal solution via an iterative way. In this article, we adopted a two-layer optimization method. Obviously, if the computational time complexity of the lower-layer optimization method is high, it is impossible to apply the two-layer optimization method in the large-scale scenarios. Moreover, as can be seen from the experimental studies in Section IV, our greedy algorithm exhibits good performance. Overall, by considering the tradeoff between the computational time complexity and accuracy, we made use of a greedy algorithm to optimize the task scheduling in the lower layer.

B. On Dynamic Environment

Although we only consider the static environment in this article, a dynamic environment can also be applied to our system. We will explain this from the following two aspects.

1) If x_i, y_i, C_i, and D_i $(i \in \mathcal{M})$ change in different time slots, then we can use ToDeTaS to jointly reoptimize the deployment of UAVs and the task scheduling in each time slot.
2) If x_i, y_i, C_i, and D_i change within a time slot, then we can tighten the delay constraints [i.e., $C7$ and $C8$ in (12)] to make the task executed in the time duration less than T.

VI. CONCLUSION

This article proposed a new multi-UAV-enabled MEC system to enhance the performance of traditional MEC systems by making use of multiple UAVs. In this system, it is necessary to jointly optimize the deployment of UAVs and task scheduling. When EAs are employed to solve this joint optimization problem, they face two issues: large-scale search space and mixed decision variables. Moreover, they usually ignore the correlation between the deployment of UAVs and task scheduling. In this article, we proposed a two-layer optimization method, called ToDeTaS, which considered the deployment of UAVs as the upper-layer optimization problem and the task scheduling as the lower-layer optimization problem. For the upper-layer optimization, a new encoding mechanism was suggested, which encoded the location of a UAV into an individual; thus, the entire population represented an entire deployment and the number of UAVs was equal to the population size. Then, DE served as the search engine and an elimination operator was designed to adaptively tune the population size. In the lower-layer optimization, for a given deployment of UAVs, we first determined the resource allocation, and then optimized the offloading decision by a greedy algorithm.

Overall, ToDeTaS has the following three advantages.

1) Compared with the original joint optimization problem, the optimization problems in the upper and lower layers have fewer decision variables, therefore reducing the dimension of the search space.
2) ToDeTaS avoids mixed decision variables by the new encoding mechanism, the elimination operator, and the derivation of resource allocation.
3) The correlation between the deployment of UAVs and the task scheduling is fully taken into consideration. Specifically, the upper layer makes the lower layer more likely to complete all tasks, and the lower layer improves the accuracy of the evaluation of the upper layer.

The performance of ToDeTaS was investigated by ten instances with up to 1000 mobile users. We also demonstrated the effectiveness of the two-layer optimization and the proposed system by various performance indicators.

The source code can be downloaded from Y. Wang's homepage: http://www.escience.cn/people/yongwang1.

Appendix

Suppose that the minimum energy to execute U_i ($i \in \mathcal{M}$) under its delay constraints in pattern k ($k \in \mathcal{K}$) is $E^{\star}_{i,k}$. Note that $E^{\star}_{i,k}$ may change with different deployments of UAVs. We are interested in identifying the maximum energy improvement for U_i in different deployments of UAVs, denoted as $\Delta E^{\star}_i$. If U_i can be executed locally and if the minimum energy to complete U_i can be improved by offloading it to a UAV, then the maximum energy improvement should be less than $E^{\star}_{i,0}$, that is, $\Delta E^{\star}_i < E^{\star}_{i,0} = \eta_1 (f^{\star}_{i,0})^{v-1} C_i$, where $f^{\star}_{i,0}$ is defined in (18). In addition, if U_i cannot be executed locally, then suppose that it is executed on UAV j. When mobile user i has the shortest distance with UAV j (i.e., mobile user i is located directly below UAV j), we can obtain the ideal minimum energy to complete U_i: $E^{\star}_{i,k,min} = P[D_i/(r_{i,k,max})] + \eta_2 (f^{\star}_{i,k})^{v-1} C_i$ ($k = j$), where $r_{i,k,max}$ represents the maximum uplink data rate and is equal to $B\log_2(1 + [(P\beta_0 G_0)/(N_0 B\theta^2((d^{MU}_{i,j,min})^2 + H^2))])$, $f^{\star}_{i,k}$ is defined in (18), and $d^{MU}_{i,j,min} = 0$. On the other hand, when mobile user i has the longest distance with UAV j (i.e., mobile user i is located on the boundary of the area covered by UAV j), we can obtain the ideal maximum energy to complete U_i: $E^{\star}_{i,k,max} = P[D_i/(r_{i,k,min})] + \eta_2 (f^{\star}_{i,k})^{v-1} C_i$ ($k = j$), where $r_{i,k,min}$ represents the minimum uplink data rate and is equal to $B\log_2(1 + [(P\beta_0 G_0)/(N_0 B\theta^2((d^{MU}_{i,j,max})^2 + H^2))])$ and $d^{MU}_{i,j,max} = H \tan\theta$. Therefore, under this condition, $\Delta E^{\star}_i = E^{\star}_{i,k,max} - E^{\star}_{i,k,min}$ ($k = j$).

According to the parameter settings in Table I, we can derive that $\sum_{i=1}^{M} \Delta E^{\star}_i < E^H$, which means that the maximum energy improvement of all tasks in different deployments of UAVs is less than the energy to hover a UAV. That is, although adding a UAV can reduce the energy to complete all tasks [i.e., the first term of the objective function of (12)], the total system energy consumption [i.e., the objective function of (12)] will definitely add. Therefore, if all tasks can be executed under delay constraints, we should use as few UAVs as possible, which is Property 1.

References

[1] J. O. B. Soh and B. C. Y. Tan, "Mobile gaming," *Commun. ACM*, vol. 51, no. 3, pp. 35–39, Mar. 2008.

[2] J. Cohen, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Las Vegas, NV, USA, 2008, pp. 5352–5355.

[3] R. Q. Hu and Y. Qian, "An energy efficient and spectrum efficient wireless heterogeneous network framework for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 94–101, May 2014.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[5] X. Sun and N. Ansari, "EdgeIoT: Mobile Edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[6] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, 2015, pp. 1–6.

[7] Z. Wu, H. Kumar, and A. Davari, "Performance evaluation of OFDM transmission in UAV wireless communication," in *Proc. 37th Southeastern Symp. Syst. Theory (SSST)*, Tuskegee, AL, USA, 2005, pp. 6–10.

[8] Y. Zhou, J. Li, L. Lamont, and C.-A. Rabbath, "Modeling of packet dropout for UAV wireless communications," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2012, pp. 677–682.

[9] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May 2016.

[10] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[11] R. Fan, J. Cui, S. Jin, K. Yang, and J. An, "Optimal node placement and resource allocation for UAV relaying network," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 808–811, Apr. 2018.

[12] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu, "Efficient 3-D placement of an aerial base station in next generation cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–5.

[13] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, Aug. 2016.

[14] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, Nov. 2017.

[15] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of UAV-mounted mobile base stations," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 604–607, Mar. 2017.

[16] V. Sharma, M. Bennis, and R. Kumar, "UAV-assisted heterogeneous networks for capacity enhancement," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1207–1210, Jun. 2016.

[17] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs," *IEEE Trans. Wireless Commun.*, vol. 15, no. 6, pp. 3949–3963, Jun. 2016.

[18] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[19] X. Lyu *et al.*, "Selective offloading in mobile edge computing for the green Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.

[20] X. Wang *et al.*, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2429–2445, Nov. 2018.

[21] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[22] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[23] J. Zhang *et al.*, "Joint offloading and resource allocation optimization for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, 2017, pp. 1–6.

[24] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," in *Proc. 27th Wireless Opt. Commun. Conf. (WOCC)*, Hualien, Taiwan, 2018, pp. 1–4.

[25] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, Jul./Sep. 2018.

[26] J. Zhang *et al.*, "Energy-latency trade-off for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[27] M. Alzenad, A. El-Keyi, and H. Yanikomeroglu, "3D placement of an unmanned aerial vehicle base station for maximum coverage of users with different QoS requirements," *IEEE Wireless Commun. Lett.*, vol. 6, no. 4, pp. 434–437, Sep. 2017.

[28] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[29] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[30] H. He, S. Zhang, Y. Zeng, and R. Zhang, "Joint altitude and beamwidth optimization for UAV-enabled multiuser communications," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 344–347, Feb. 2018.

[31] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[32] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[33] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, Aug. 2008.

[34] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[35] T. Liao, K. Socha, M. A. Montes De Oca, T. Stutzle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 503–518, Aug. 2014.

[36] B. Hutt and K. Warwick, "Synapsing variable-length crossover: Meaningful crossover for variable-length genomes," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 118–131, Feb. 2007.

[37] Y. Wang, H. Liu, H. Long, Z. Zhang, and S. Yang, "Differential evolution with a new encoding mechanism for optimizing wind farm layout," *IEEE Trans. Ind. Inf.*, vol. 14, no. 3, pp. 1040–1054, Mar. 2018.

[38] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[39] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem," *Math. Program.*, vol. 8, no. 1, pp. 91–103, Dec. 1975.

[40] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, 2018, pp. 1–6.

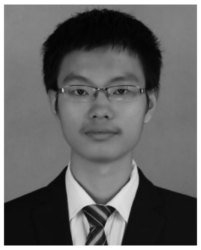
[41] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[42] P. Kadlec and V. Šeděnka, "Particle swarm optimization for problems with variable number of dimensions," *Eng. Optim.*, vol. 50, no. 3, pp. 382–399, Mar. 2018.

Yong Wang (M'08–SM'17) received the Ph.D. degree in control science and engineering from Central South University, Changsha, China, in 2011.

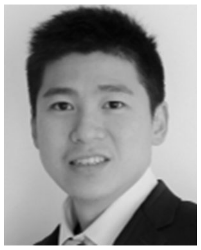
He is a Professor with the School of Automation, Central South University. His current research interests include theory, algorithm design, and interdisciplinary applications of computational intelligence.

Prof. Wang was a Web of Science Highly Cited Researcher in Computer Science in 2017 and 2018. He is an Associate Editor of the *Swarm and Evolutionary Computation*.



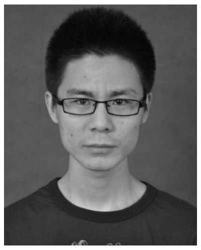
Zhi-Yang Ru received the B.S. degree in automation from Xiangtan University, Xiangtan, China, in 2016. He is currently pursuing the M.S. degree in control science and engineering with Central South University, Changsha, China.

His current research interests include evolutionary computation and mobile edge computing.



Kezhi Wang (M'15) received the B.E. and M.E. degrees in automation from Chongqing University, Chongqing, China, in 2008 and 2011, respectively, and the Ph.D. degree in engineering from the University of Warwick, Coventry, U.K., in 2015.

He was a Senior Research Officer with the University of Essex, Colchester, U.K. He is currently a Lecturer with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. His current research interests include wireless communication, mobile edge computing, and artificial intelligence.



Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in control science and engineering with the Central South University, Changsha, China.

His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.