Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system*

Pei-qiu HUANG[1], Yong WANG[‡1], Ke-zhi WANG[2]

[1]*School of Automation, Central South University, Changsha 410083, China*

[2]*Department of Computer and Information Sciences, Northumbria University, Newcastle NE1 8ST, UK*

E-mail: pqhuang@csu.edu.cn; ywang@csu.edu.cn; kezhi.wang@northumbria.ac.uk

Received July 2, 2020; Revision accepted Sept. 9, 2020; Crosschecked Oct. 29, 2020

Abstract: We study a mobile edge computing system assisted by multiple unmanned aerial vehicles (UAVs), where the UAVs act as edge servers to provide computing services for Internet of Things devices. Our goal is to minimize the energy consumption of this system by planning the trajectories of UAVs. This problem is difficult to address because when planning the trajectories, we need to consider not only the order of stop points (SPs), but also their deployment (including the number and locations) and the association between UAVs and SPs. To tackle this problem, we present an energy-efficient trajectory planning algorithm (TPA) which comprises three phases. In the first phase, a differential evolution algorithm with a variable population size is adopted to update the number and locations of SPs at the same time. In the second phase, the k-means clustering algorithm is employed to group the given SPs into a set of clusters, where the number of clusters is equal to that of UAVs and each cluster contains all SPs visited by the same UAV. In the third phase, to quickly generate the trajectories of UAVs, we propose a low-complexity greedy method to construct the order of SPs in each cluster. Compared with other algorithms, the effectiveness of TPA is verified on a set of instances at different scales.

Key words: Multiple unmanned aerial vehicles; Mobile edge computing; Trajectory planning; Differential evolution; k-means clustering algorithm; Greedy method

https://doi.org/10.1631/FITEE.2000315 **CLC number:** TN929.5; TP301.6

1 Introduction

With the development of mobile communication technology and the popularization of Internet of Things (IoT) devices, a considerable number of resource-intensive applications are emerging, such as face recognition, virtual reality, and online games (Xu et al., 2018; Zhang J et al., 2019). Despite the growing capabilities of IoT devices, their computing and battery capacities remain insufficient due to physical size limitations. Therefore, it is a challenge to execute resource-intensive tasks on IoT devices.

Mobile edge computing (MEC) is recognized as a promising technology to address the above challenge. It provides computing services to IoT devices by offloading tasks to edge servers at the edge of the network (Jin et al., 2019; Wang KZ et al., 2019; Huang PQ et al., 2020a). In this way, MEC can reduce latency and energy consumption during task execution. However, MEC still has some limitations. For example, the locations of edge servers are usually fixed and cannot be adjusted according to user requirements. In addition, in large-scale natural disasters, the existing terrestrial communication networks could be destroyed, in which case it would be difficult for MEC to provide timely services (Mozaffari et al.,

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61673397 and 61976225) and the Fundamental Research Funds for the Central Universities of Central South University, China (No. 2020zzts129)

ORCID: Pei-qiu HUANG, https://orcid.org/0000-0001-6278-4566; Yong WANG, https://orcid.org/0000-0001-7670-3958

2019).

Unmanned aerial vehicles (UAVs), due to their autonomy and flexibility, have been widely used in various fields (Low et al., 2019; Zollars et al., 2019; Zaini and Xie, 2020). Recently, some attempts have been made to use UAVs to enhance the capabilities of MEC systems. Zhang L et al. (2018) explored the energy-aware dynamic resource allocation problem for a UAV-assisted MEC system over Internet of vehicles. Du et al. (2019) optimized joint resource and workflow scheduling in a UAV-enabled wirelessly powered MEC system. Garg et al. (2018) investigated the application of a UAV-empowered MEC system in cyber-threat detection of smart vehicles. In addition, to take full advantage of high mobility of a UAV, some researchers have focused on trajectory planning in UAV-assisted MEC systems. For instance, Diao et al. (2019) optimized joint trajectory and data allocation to minimize the energy consumption. Jeong et al. (2018) studied bit allocation and trajectory planning under latency and energy budget constraints. Hu et al. (2019) developed a UAV-assisted relaying and MEC system, where the UAV can act as an MEC server or a relay, and then they proposed a joint task scheduling and trajectory optimization algorithm to minimize the weighted sum energy consumption of the system.

However, the above-mentioned studies considered only single-UAV-assisted MEC systems. In fact, collaboration among multiple UAVs can improve the capability of such systems (Chen J et al., 2016). Therefore, some researchers have studied multi-UAV-assisted MEC systems, where a group of UAVs, rather than a single UAV, act as edge servers to provide computing services for IoT devices (Li et al., 2020; Zhang J et al., 2020). For example, Yang et al. (2019) optimized the power consumption of a multi-UAV-assisted MEC system by considering the joint device association, power control, computing capacity allocation, and location planning. Wang Y et al. (2020) designed a two-layer optimization algorithm for joint UAV deployment and task scheduling in a multi-UAV-assisted MEC system. Chen WH et al. (2019) investigated the quality of service in a multi-UAV-assisted MEC system.

In this study, we investigate the trajectory planning problem in a multi-UAV-assisted MEC system. Compared with conventional trajectory planning problems, e.g., traveling salesman problems (Huang L et al., 2017) and vehicle routing problems (Wang JH et al., 2016), the studied problem is more challenging due to the fact that the deployment of the stop points (SPs) of UAVs is unknown a priori. In addition, different from trajectory planning problems in single-UAV-assisted MEC systems, in the case of multiple UAVs, we need to consider the association between UAVs and SPs. That is, for a given SP, we need to assign a specific UAV to visit it. The main contributions of this paper are summarized as follows:

1. A trajectory planning problem in a multi-UAV-assisted MEC system is formulated with the aim of minimizing the energy consumption of the system by considering the deployment (including the number and locations) of SPs, the association between UAVs and SPs, and the order of SPs.

2. An energy-efficient trajectory planning algorithm, called trajectory planning algorithm (TPA), is proposed to tackle the trajectory planning problem. TPA consists of three phases. First, a differential evolution (DE) algorithm with a variable population is adopted to optimize the deployment of SPs. Subsequently, the k-means clustering algorithm is used to group the given SPs into several clusters, SPs in each of which are associated with the same UAV to be visited. Finally, a greedy method is proposed to construct the order of SPs in each cluster.

3. Extensive experiments are carried out on a set of instances with up to 200 IoT devices. The experimental results demonstrate that TPA achieves better performance compared with other algorithms.

2 System model and problem formulation

As shown in Fig. 1, we consider a multi-UAV-assisted MEC system involving n IoT devices (denoted as $\mathcal{N} = \{1, 2, \ldots, n\}$) and m rotary-wing UAVs with edge servers (denoted as $\mathcal{M} = \{1, 2, \ldots, m\}$). In this system, each IoT device has a resource-intensive task to be completed. For simplification, the i^{th} task (we refer to the task of the i^{th} IoT device as the i^{th} task) is expressed as a two-tuple (D_i, S_i), where D_i and S_i denote the size of the input data of the i^{th} task and the computing resource required to complete a single bit in the i^{th} task, respectively. Due to the limited computing capacity, these tasks are first offloaded to the MEC servers, and then their

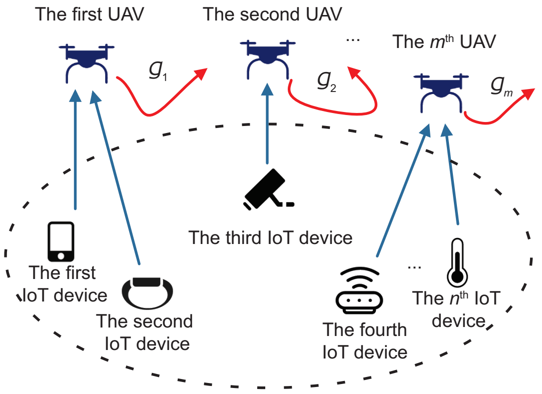

Fig. 1 A multi-UAV-assisted MEC system involving m rotary-wing UAVs and n IoT devices

results are returned to the IoT devices after the computation is completed.

In this study, the UAVs can change their SPs to reduce the distance from the IoT devices. We define the set of SPs of the j^{th} UAV as $\mathcal{K}_j = \{1, 2, \ldots, k_j\}$, where k_j is the number of SPs of the j^{th} UAV and it is unknown a priori. Moreover, the trajectory of the j^{th} UAV is represented as a sequence of SPs in $\mathcal{K}_j$: $\mathcal{G}_j = \{(X_{j1}, Y_{j1}), (X_{j2}, Y_{j2}), \ldots, (X_{jk_j}, Y_{jk_j})\}$, where (X_{jl}, Y_{jl}) $(l \in \mathcal{K}_j)$ denotes the location of the l^{th} SP of the j^{th} UAV. Like Huang PQ et al. (2020b) and Wang Y et al. (2020), we assume that UAVs fly at a fixed altitude H, and therefore we show only the values on the x and y axes. In addition, all SPs in $\mathcal{G}_j$ are visited by the j^{th} UAV one by one, where the first SP is visited first and the k_j^{th} SP is visited last.

We assume that the i^{th} IoT device is located at $(x_i, y_i, 0)$. Therefore, the distance between the i^{th} IoT device and the j^{th} UAV at the l^{th} SP is expressed as

$$\begin{aligned} d_{ijl} = &\sqrt{(x_i - X_{jl})^2 + (y_i - Y_{jl})^2 + H^2}, \\ &\forall i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j. \end{aligned} \tag{1}$$

To reduce the transmission time and energy consumption, IoT devices always send their tasks to the closest SP. We define variable a_{ijl} to represent the association between the i^{th} IoT device and the j^{th} UAV at the l^{th} SP. Specifically, $a_{ijl} = 1$ if the i^{th} IoT device is served by the j^{th} UAV at the l^{th} SP, and $a_{ijl} = 0$ otherwise. Thus, one can obtain

$$\text{C1}: a_{ijl} = \begin{cases} 1, \text{ if } (j, l) = \underset{j \in \mathcal{M}, l \in \mathcal{K}_j}{\arg\min}\ d_{ijl}, \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Since each task cannot be further divided into subtasks, the following constraint should be satisfied:

$$\text{C2}: \sum_{j=1}^{m} \sum_{l=1}^{k_j} a_{ijl} = 1,\ \forall i \in \mathcal{N}. \tag{3}$$

Due to the bandwidth limitation, the j^{th} UAV at the l^{th} SP can simultaneously serve at most M IoT devices. Thus, one has

$$\text{C3}: \sum_{i=1}^{n} a_{ijl} \le M,\ \forall j \in \mathcal{M}, l \in \mathcal{K}_j. \tag{4}$$

Moreover, each UAV at each SP serves at least one IoT device, and thus the total number of SPs of all UAVs, denoted as k, should satisfy the following constraint:

$$\text{C4}:\ k_{\min} \le k \le k_{\max}, \tag{5}$$

where $k = \sum_{j=1}^{m} k_j$, $k_{\min} = \lfloor \frac{n}{M} \rfloor$ (Here, $\lfloor \cdot \rfloor$ denotes the rounding down operator), and $k_{\max} = n$.

The transmission rate of the i^{th} IoT device for sending data to the j^{th} UAV at the l^{th} SP is expressed as

$$r_{ijl} = B\log_2\left(1 + \frac{p_i^{\text{t}} h_0}{\sigma^2 d_{ijl}^2}\right), \forall i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j, \tag{6}$$

where p_i^{t} denotes the transmitting power of the i^{th} IoT device, h_0 denotes the channel power gain at the reference distance $d_0 = 1$ m, σ^2 denotes the white Gaussian noise power, and B denotes the bandwidth.

The transmission time and energy consumption of the i^{th} IoT device for sending data to the j^{th} UAV at the l^{th} SP are given by

$$T_{ijl}^{\text{t}} = \frac{D_i}{r_{ijl}},\ \forall i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j \tag{7}$$

and

$$E_{ijl}^{\text{t}} = p_i^{\text{t}} T_{ijl}^{\text{t}} = \frac{p_i^{\text{t}} D_i}{r_{ijl}},\ \forall i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j. \tag{8}$$

The whole energy consumption of all IoT devices is expressed as (due to the fact that the size of the output results is smaller than that of the input data of the task, we omit the transmission time and energy consumption of the output results)

$$E_{\text{IoT}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=1}^{k_j} a_{ijl} E_{ijl}^{\text{t}}. \tag{9}$$

After receiving the input data, UAVs start to execute the tasks. Given the computing resource c_{ijl}, the computing time of the i^{th} task on the j^{th} UAV at the l^{th} SP can be obtained by

$$T_{ijl}^{\text{c}} = \frac{D_i S_i}{c_{ijl}},\ \forall i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j. \tag{10}$$

In fact, the j^{th} UAV will not move to the next SP until all tasks sent to the l^{th} SP have been completed. Therefore, the hovering time of the j^{th} UAV at the l^{th} SP is equal to the maximum execution time of all tasks (i.e., the sum of the transmission time and computing time), which is given by

$$T_{jl}^{\text{h}} = \max_{i \in \mathcal{N}} \{a_{ijl}(T_{ijl}^{\text{t}} + T_{ijl}^{\text{c}})\},\ \forall j \in \mathcal{M}, l \in \mathcal{K}_j. \tag{11}$$

Then, the hovering energy consumption of the j^{th} UAV is given by

$$E_j^{\text{h}} = \sum_{l=1}^{k_j} p^{\text{h}} T_{jl}^{\text{h}},\ \forall j \in \mathcal{M}, \tag{12}$$

where p^{h} is the hovering power of the UAV.

Furthermore, given the trajectory of the j^{th} UAV (i.e., $\mathcal{G}_j$), the flight time and energy consumption are expressed as

$$T_j^{\text{f}} = \frac{1}{v} \sum_{l=2}^{k_j} \sqrt{(X_{jl} - X_{j(l-1)})^2 + (Y_{jl} - Y_{j(l-1)})^2} \tag{13}$$

and

$$E_j^{\text{f}} = p^{\text{f}} T_j^{\text{f}}, \tag{14}$$

where $\forall j \in \mathcal{M}$, v is the flight speed of the UAV, and p^{f} is the flight power of the UAV.

The whole energy consumption of all UAVs consists of the hovering energy consumption and flight energy consumption (compared with the hovering energy consumption and flight energy consumption of UAVs, the computing energy consumption of the UAVs is smaller (Wang L et al., 2019); thus, we omit the computing energy consumption of UAVs), which can be expressed as

$$E_{\text{UAV}} = \sum_{j=1}^{m} \left(E_j^{\text{h}} + E_j^{\text{f}}\right). \tag{15}$$

In this study, we aim to optimize the trajectories of UAVs (i.e., $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m$) to minimize the energy consumption of the system consisting of UAVs and IoT devices. Thus, the problem can be formulated as

$$\begin{aligned}
\min_{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m} \ & (E_{\text{UAV}} + \alpha E_{\text{IoT}}) \\
\text{s.t.} \ & \text{C1}: a_{ijl} \in \{0, 1\},\ \forall i \in \mathcal{N},\ j \in \mathcal{M},\ l \in \mathcal{K}_j, \\
& \text{C2}: \sum_{j=1}^{m} \sum_{l=1}^{k_j} a_{ijl} = 1,\ \forall i \in \mathcal{N}, \\
& \text{C3}: \sum_{i=1}^{n} a_{ijl} \leq M,\ \forall j \in \mathcal{M},\ l \in \mathcal{K}_j, \\
& \text{C4}: k_{\min} \leq k \leq k_{\max}, \\
& \text{C5}: X_{\min} \leq X_{jl} \leq X_{\max},\ \forall j \in \mathcal{M},\ l \in \mathcal{K}_j, \\
& \text{C6}: Y_{\min} \leq Y_{jl} \leq Y_{\max},\ \forall j \in \mathcal{M},\ l \in \mathcal{K}_j,
\end{aligned} \tag{16}$$

where $\alpha \geq 0$ is the weight parameter between the energy consumption of UAVs and IoT devices, $X_{\min}$ and $X_{\max}$ are the lower and upper bounds of X_{jl}, respectively, and $Y_{\min}$ and $Y_{\max}$ are the lower and upper bounds of Y_{jl}, respectively.

3 The proposed approach

3.1 Framework of the proposed approach

By analyzing problem (16), there are two challenges that need to be considered:

1. To address problem (16), we need to know how many SPs are suitable and where they are located, which UAV is assigned to visit the given SP, and how to visit SPs in turn for each UAV. Therefore, problem (16) can be decomposed into three sub-problems: the deployment (including the number and locations) of SPs, the association between UAVs and SPs, and the order of SPs. However, the deployment of SPs, the association between UAVs and SPs, and the order of SPs are coupled with each other. Specifically, the association between UAVs and SPs can be determined only after the deployment of SPs is obtained. Moreover, the order of SPs can be constructed only after the association between UAVs and SPs is determined. Therefore, if they are optimized at the same time, it may lead to a poor performance.

2. Since the number of SPs is unknown when optimizing their deployment, the gradient information is not available. As a result, traditional gradient-based methods cannot optimize the deployment of SPs. As a class of gradient-free optimization

methods, evolutionary algorithms (EAs) have potential to optimize the deployment of SPs (Zhang J et al., 2019). However, in EAs, each individual typically represents an entire deployment. Due to the fact that the number of SPs is unknown a priori, the length of the individual is not fixed. However, the commonly used crossover and mutation operators are designed for fixed-length individuals (Ryerkerk et al., 2019). Therefore, using conventional EAs directly would be ineffective in optiming the deployment of SPs.

To this end, we propose a trajectory planning algorithm, called TPA, which has the following technical advantages:

1. Considering the strong coupling among the deployment of SPs, the association between UAVs and SPs, and the order of SPs, TPA plans the trajectories of UAVs at each iteration through three phases: update of the deployment of SPs, generation of the association between UAVs and SPs, and construction of the order of SPs.

2. As shown in Fig. 2, in TPA, each individual represents the location of an SP; thus, the population represents a whole deployment, rather than a set of deployments. Since the lengths of individuals are the same (i.e., two), we can directly adopt the commonly used crossover and mutation operators to update the deployment of SPs.

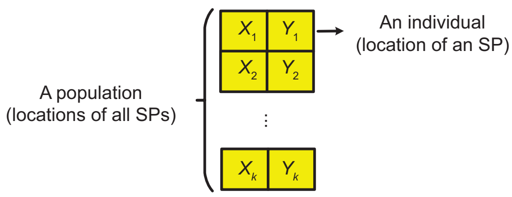

Fig. 2 Encoding mechanism used in this study

The framework of TPA is presented in Algorithm 1. In the initialization, locations of SPs of all UAVs are produced randomly, forming an initial population $\mathcal{P}=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_{k_{\max}},Y_{k_{\max}})\}$ (line 3). Subsequently, the association between UAVs and SPs in $\mathcal{P}$ is determined using the k-means clustering algorithm (which will be described later), and the order of SPs for each UAV is constructed using the greedy method (which will be described later). After that, $\mathcal{P}$ is evaluated via Eq. (16) (line 6). If $\mathcal{P}$ is feasible, the initialization is produced successfully; otherwise, it is repeated until $\mathcal{P}$ is feasible or the number of fitness evaluations

Algorithm 1 Framework of trajectory planning algorithm

1: FEs=0
2: **repeat**
3:   Produce randomly an initial population $\mathcal{P}$
4:   Determine the association between UAVs and SPs in $\mathcal{P}$ via the k-means clustering algorithm
5:   Construct the order of SPs of each UAV via the greedy method
6:   Evaluate $\mathcal{P}$ via Eq. (16)
7:   FEs=FEs+1
8: **until** $\mathcal{P}$ is feasible or FEs$\geq$MaxFEs
9: **while** FEs$<$MaxFEs **do**
10:   Produce an offspring population $\mathcal{Q}$ via "DE/rand/1" and the binomial operator of DE
11:   **for** $i=1:|\mathcal{Q}|$ **do**
12:     Construct three new populations $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ via Algorithm 2
13:     **for** $l=1:3$ **do**
14:       Determine the association between SPs in $\mathcal{P}_l$ and UAVs via the k-means clustering algorithm
15:       Construct the order of SPs of each UAV via the greedy method
16:     **end for**
17:     Evaluate $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ via Eq. (16)
18:     FEs=FEs+3
19:     **if** at least one feasible population exists among $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ **then**
20:       Update $\mathcal{P}$ by the feasible population among $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ with the greatest performance improvement against $\mathcal{P}$
21:     **end if**
22:   **end for**
23: **end while**

(FEs) is not smaller than MaxFEs, where MaxFEs denotes the maximum number of FEs. During the evolution, an offspring population $\mathcal{Q}$ is first produced via "DE/rand/1" and the binomial operator of DE (line 10). Subsequently, three new populations $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are constructed (line 12) via Algorithm 2. Then, SPs in $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are associated with UAVs (line 14) via the k-means clustering algorithm in Algorithm 3, and the order of SPs for each UAV is constructed (line 15) via the greedy algorithm in Algorithm 4. Afterward, we evaluate $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ via Eq. (16) (line 17). Finally, the feasible population among $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ with the greatest performance improvement against $\mathcal{P}$ is used to replace $\mathcal{P}$ if at least one feasible population exists among $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ (lines 19–21). The evolution continues until FEs≥MaxFEs. Fig. 3 illustrates the framework of TPA.

Algorithm 2 Generation of three new populations

```
P_1 ← Insert the i^th individual in Q to P
P_2 ← Replace a random individual in P by the i^th individual in Q
P_3 ← Delete a random individual in P
```

Algorithm 3 k-means clustering algorithm for the clustering of SPs

```
Initialize C_j = ∅ (∀j ∈ M)
Randomly select an SP for each cluster
repeat
    for i = 1 : k do
        for j = 1 : m do
            Calculate the distance d_ij from the i^th SP to the center of all SPs in the j^th cluster
        end for
        j' = arg min_{j∈M} d_ij
        Add the i^th SP into C_j'
    end for
until the center of SPs in each cluster is no longer changed
Associate the j^th UAV with SPs in C_j (∀j ∈ M)
```

Algorithm 4 Greedy method for constructing the order of SPs

```
for j = 1 : m do
    Select the location of a random SP from C_j as the current SP of the j^th UAV
    for l = 1 : k_j do
        Move the current SP from C_j into G_j
        Calculate the distances from the current SP to all SPs in C_j
        Find the closest SP from the current SP as the new current SP
    end for
end for
Output G_j (j ∈ M)
```

3.2 Update of the deployment of SPs

Updating the deployment of SPs consists of two parts: the locations and the number of SPs. In TPA, DE (Storn and Price, 1997) is used to update the locations of SPs. The reason is that DE is a simple and effective EA and has been successfully applied in many fields (Xin et al., 2012; Wang BC et al., 2018). Specifically, we first use "DE/rand/1" and the binomial operator (Wang Y et al., 2011) of DE to produce an offspring population $\mathcal{Q}$ consisting of the locations of new SPs, and then adopt the individuals in $\mathcal{Q}$ to update $\mathcal{P}$. In this way, the locations of SPs can be updated.

Since the location of each SP is treated as an individual in DE, the whole population represents the locations of all SPs. Therefore, the population size is equal to the number of SPs. To update the number of SPs, the population size should be variable during the evolution. In other words, the population size can be increased, kept unchanged, or reduced. As a result, we first construct three populations of different sizes via Algorithm 2. Specifically, for the i^{th} individual in $\mathcal{Q}$, a new population $\mathcal{P}_1$ is constructed by incorporating it into $\mathcal{P}$ and another new population $\mathcal{P}_2$ is constructed by using it to replace a random individual in $\mathcal{P}$. In addition, the third new population $\mathcal{P}_3$ is constructed by removing a random individual from $\mathcal{P}$. It is clear that the population sizes of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are larger than, the same as, and smaller than that of $\mathcal{P}$, respectively. Therefore, when $\mathcal{P}_1$, $\mathcal{P}_2$, or $\mathcal{P}_3$ is selected to update $\mathcal{P}$, the population size will be increased, kept unchanged, or reduced, respectively. In this way, the number of SPs can be updated.

3.3 Generation of the association between UAVs and SPs

After generating a new population, we need to determine the association between UAVs and SPs. That is, these SPs are assigned to UAVs to be visited. In this study, the k-means clustering algorithm (Jain, 2010) is used to group SPs into m clusters, where SPs in each cluster are visited by the same UAV. The loss function of the k-means clustering algorithm is given as

$$\min_{\mathcal{C}_j, j\in\mathcal{M}} \sum_{j\in\mathcal{M}} \sum_{(X_l,Y_l)\in\mathcal{C}_j} \sqrt{(X_l-\hat{X}_j)^2+(Y_l-\hat{Y}_j)^2}, \tag{17}$$

where $\hat{X}_j = \frac{1}{|\mathcal{C}_j|}\sum_{(X_l,Y_l)\in\mathcal{C}_j} X_l$ and $\hat{Y}_j = \frac{1}{|\mathcal{C}_j|}\sum_{(X_l,Y_l)\in\mathcal{C}_j} Y_l$.

From function (17), we can find that the k-means clustering algorithm can group the closely spaced SPs into the same cluster. Since SPs in the same cluster are visited by the same UAV, the flying

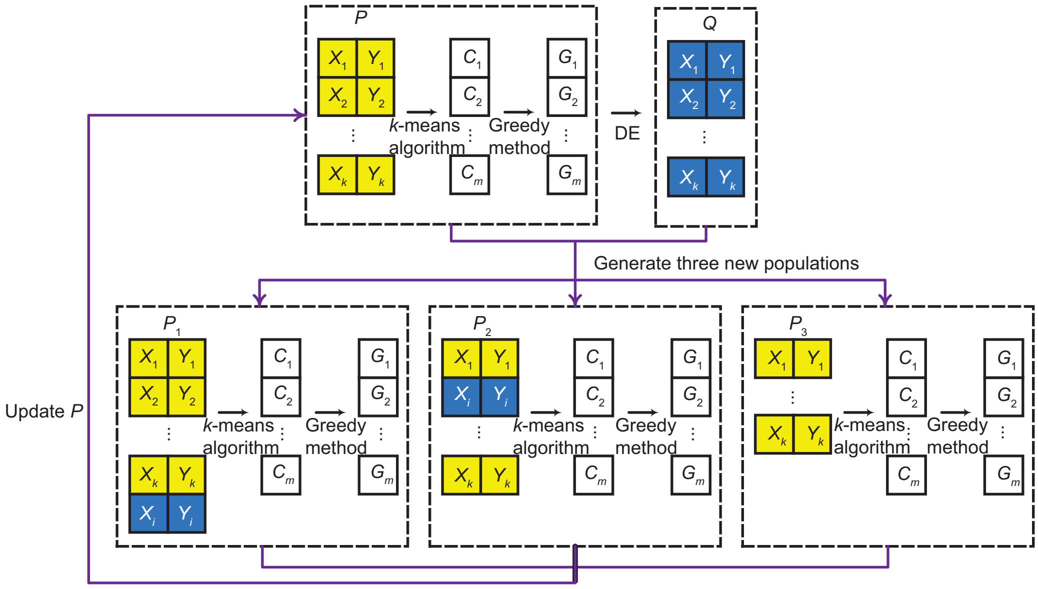

Fig. 3 Overall framework of trajectory planning algorithm

distance of the UAVs can be reduced, thereby reducing energy consumption in the system.

Algorithm 3 presents the procedure for the k-means clustering algorithm for the association between UAVs and SPs. First, we initialize m clusters $\mathcal{C}_j = \varnothing$ $(\forall j \in \mathcal{M})$, and then randomly select an SP for each cluster (lines 1 and 2). Afterward, we calculate the distance from each SP to the center of SPs in each cluster and add the SP into the nearest cluster (lines 4–10). The above procedure is repeated until the center of SPs in each cluster is no longer changed. Finally, all SPs in $\mathcal{C}_j$ $(\forall j \in \mathcal{M})$ are associated with the j^{th} UAV.

3.4 Construction of the order of SPs

In this subsection, we construct the order of SPs for each UAV to minimize the flying distance of UAVs. In fact, this problem is essentially a traveling salesman problem. Although classical mathematical programming methods (such as the branch and bound algorithm) and the population-based methods (such as the ant colony algorithm and genetic algorithm (GA)) have been successfully adopted to address traveling salesman problems, they suffer from high computational time complexity. To this end, we propose a low-complexity greedy method for constructing the order of SPs.

As shown in Algorithm 4, for the first UAV, we select a random SP from $\mathcal{C}_1$ as the current SP (line 2). Subsequently, the current SP is moved from $\mathcal{C}_1$ into $\mathcal{G}_1$ (line 4). The distances from the current SP to all SPs in $\mathcal{C}_1$ are then calculated, and the closest SP from the current SP is chosen as the new current SP (lines 5 and 6). The above procedure is repeated until $\mathcal{C}_1$ is empty. As a result, the trajectory of the first UAV (i.e., $\mathcal{G}_1$) is generated. The remaining UAVs experience the above process one by one.

Remark 1 In the existing studies on trajectory planning problems in multi-UAV-assisted MEC systems (Li et al., 2020; Zhang J et al., 2020), it is assumed that all UAVs have the same working time. In addition, the working time is divided into a series of time slots in a discretized manner, and then the SP of each UAV is determined for each time slot. In this case, all UAVs have the same number of SPs. However, we do not assume that all UAVs have the same working time and the same number of SPs.

4 Experimental study

The parameter settings of the studied multi-UAV-assisted MEC system are summarized as follows: We assume that the IoT devices were distributed randomly in a 1000 m×1000 m square region; there were four UAVs flying at a height of 200 m at a speed of 20 m/s; D_i $(i \in \mathcal{N})$ was distributed randomly in $[1, 10^3]$ MB; S_i $(i \in \mathcal{N})$ was set to 100 cycles/bit; c_{ijl} $(i \in \mathcal{N}, j \in \mathcal{M}, l \in \mathcal{K}_j)$ was set to 10 GHz; M was set to five; p_i^{t} was set to 0.1 W; p^{h} and p^{f} were set to 1000 W. In addition, σ^2 was set to -174 dBm; h_0 was set to -30 dB; B was set to 1 MHz; α was set to 10 000. In this study, we adopted eight instances with different numbers of IoT devices to evaluate the performance of TPA: $n = 60, 80, 100, 120, 140, 160, 180, 200$. The parameters of TPA were set as follows: $F = 0.6$, CR=0.5, and MaxFEs = 50 000. Each algorithm was executed independently 20 runs for each instance. Moreover, to test the statistical significance between TPA and each competitor, the Wilcoxon rank-sum test at a 0.05 significance level was conducted. In the experimental results, "↑," "≈," and "↓" represent that TPA performed significantly better than, equivalent to, and worse than its competitor, respectively. We implemented all the experiments in MATLAB and tested them on a personal computer running with an Intel Core i5-7500 CPU @3.40 GHz and 8 GB of RAM.

4.1 Effectiveness of the deployment of SPs

TPA adopts DE with a variable population to update the deployment of SPs. To verify the effectiveness of the deployment of SPs, we replace DE used in TPA with three algorithms separately, VLGA (Ting et al., 2009), JGGA (Chan et al., 2007), and DEEM (Wang Y et al., 2018), resulting in three new algorithms: TPA-VLGA, TPA-JGGA, and TPA-DEEM. In VLGA, the uniform and cut-and-splice crossover operators are used to produce variable-length individuals. JGGA employs continuous auxiliary variables ranging from 0 to 1 to control the expression of the locations of SPs. If the auxiliary variable is larger than 0.5, the corresponding SP is used; otherwise, it is not used. DEEM develops an encoding mechanism similar to that used in this study, but it needs to set the number of SPs in advance. In this study, we preset the number of SPs in DEEM to a random value in $[k_{\min}, k_{\max}]$.

Table 1 presents the experimental results of TPA and three comparators regarding the average and standard deviation of energy consumption (EC) over 20 runs. The statistical test results between TPA and each of the three competitors are summarized at the bottom of Table 1. Note that, if not all the IoT devices are served in one run, the run was considered to be infeasible. In this case, we give only the feasibility rate in Table 1. It is clear that TPA-VLGA, TPA-JGGA, and TPA can achieve a 100% feasibility rate on each instance. However, TPA shows better performance than TPA-VLGA and TPA-JGGA on each instance in terms of the average EC. As for DEEM, it cannot achieve a 100% feasibility rate on any instance. In addition, TPA is significantly better than each of the three competitors on all instances. Fig. 4 shows the evolution of the average EC of TPA-VLGA, TPA-JGGA, and TPA when $n = 160, 180$, and 200. Since TPA-DEEM cannot achieve a 100% feasibility rate on these instances, the evolution of the average EC of TPA-DEEM is not presented. It can be seen that TPA provides the best performance in all algorithms. Moreover, we present the average running time of TPA-VLGA, TPA-JGGA, and TPA on each instance in Fig. 5. Although the difference among the average running time of TPA-VLGA, TPA-JGGA, and TPA is small, it can still be found that TPA-VLGA on three instances, TPA-JGGA on one instance, and TPA on four instances require less running time.

Table 1 Experimental results of TPA-VLGA, TPA-JGGA, TPA-DEEM, and TPA in terms of average energy consumption (EC) (J) over 20 runs

n	Mean (standard deviation)			Feasibility rate
	TPA-VLGA	TPA-JGGA	TPA	TPA-DEEM
60	1.57e + 6 (2.33e + 4) ↑	1.53e + 6 (2.47e + 4) ↑	1.40e + 6 (2.03e + 4)	90% ↑
80	2.36e + 6 (4.20e + 4) ↑	2.22e + 6 (2.33e + 4) ↑	2.06e + 6 (2.68e + 4)	95% ↑
100	3.07e + 6 (3.41e + 4) ↑	2.94e + 6 (2.79e + 4) ↑	2.68e + 6 (3.73e + 4)	90% ↑
120	3.28e + 6 (3.54e + 4) ↑	3.12e + 6 (2.74e + 4) ↑	2.82e + 6 (6.29e + 4)	80% ↑
140	4.31e + 6 (4.39e + 4) ↑	4.09e + 6 (3.59e + 4) ↑	3.71e + 6 (3.03e + 4)	70% ↑
160	5.03e + 6 (6.89e + 4) ↑	4.77e + 6 (2.59e + 4) ↑	4.21e + 6 (5.21e + 4)	75% ↑
180	5.63e + 6 (6.06e + 4) ↑	5.39e + 6 (3.87e + 4) ↑	4.83e + 6 (4.17e + 4)	85% ↑
200	6.27e + 6 (1.00e + 5) ↑	6.07e + 6 (3.86e + 4) ↑	5.35e + 6 (4.20e + 4)	80% ↑
↑ / ↓ / ≈	7/0/0	7/0/0		7/0/0

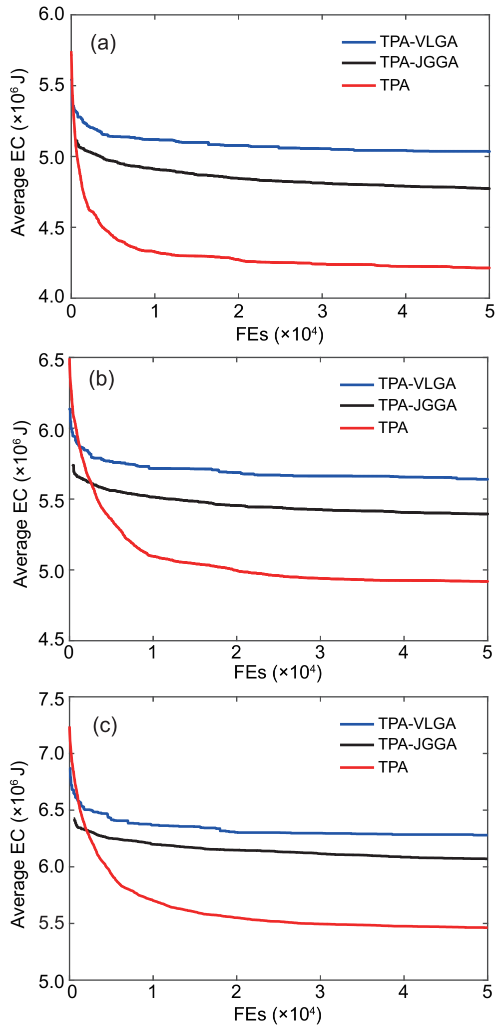

Fig. 4 Evolution of the average energy consumption (EC) obtained by TPA-VLGA, TPA-JGGA, and TPA on three instances: (a) n = 160; (b) n = 180; (c) n = 200

References to color refer to the online version of this figure

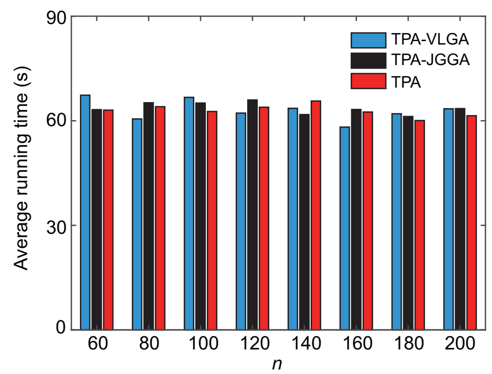

Fig. 5 Average running time of TPA-VLGA, TPA-JGGA, and TPA on each instance

References to color refer to the online version of this figure

The above-mentioned phenomenon is attributed mainly to the following reason: Due to the different lengths of individuals, TPA-VLGA searches for the optimal deployment of SPs in a variable-dimensional space, which may cause a confused search. Although individuals in TPA-JGGA are of the same length, the introduction of auxiliary variables leads to an increase in the length of the individuals, thus encountering the curse of dimensionality, especially in large-scale instances. Since the number of SPs needs to be set in advance, TPA-DEEM cannot update the number of SPs during evolution. Note that an inappropriate number of SPs may result in not all IoT devices being served. Since TPA can simultaneously update the number and locations of SPs and the lengths of individuals are the same and are low, it can achieve a better performance.

4.2 Effectiveness of the association between UAVs and SPs

To verify the effectiveness of the association between UAVs and SPs, we develop a variant of TPA without the k-means clustering algorithm, called TPA-WoK, in which the UAVs are randomly associated with SPs. Table 2 presents the average and standard deviation of EC over 20 runs, as well as the statistical results between TPA and TPA-WoK. It is clear that TPA outperforms TPA-WoK on all instances in terms of the average EC. In addition, TPA presents significantly better statistical test results on all instances. The reason is given as follows: From Figs. 6a and 6b, we can observe that TPA can associate the closely spaced SPs with the same UAV, but TPA-WoK cannot. Therefore, TPA can reduce the flight distance of UAVs to lower the energy consumption of the system, which verifies the effectiveness of the association between UAVs and SPs.

4.3 Effectiveness of the order of SPs

In this subsection, we investigate the effectiveness of the order of SPs by comparing TPA with two variants, called TPA-RAN and TPA-GA. TPA-RAN randomly generates the order of SPs, while TPA-GA employs GA to optimize the order of SPs. Note that in TPA-GA, a population of 10 individuals is used to search for the optimal trajectory for each UAV and the maximum number of iterations is set to 50. Table 2 presents the experimental results of TPA-RAN, TPA-GA, and TPA. It is clear that TPA performs better than TPA-RAN and TPA-GA. To further validate the effectiveness of the order of SPs, we present the trajectories of UAVs obtained by TPA-RAN and TPA-GA in Figs. 6c and 6d. Compared with TPA, TPA-RAN and TPA-GA obtain longer flight trajectories, resulting in higher energy consumption. The above experimental results verify the effectiveness of the order of SPs. The poor performance of TPA-GA could appear confusing. It is explained as follows: As shown in Fig. 7, the average running time of TPA-GA is longer than that of TPA due to the

Table 2 Experimental results of TPA-WoK, TPA-RAN, TPA-GA, and TPA in terms of average energy consumption (EC) (J) over 20 runs

n	Mean (standard deviation)			
	TPA-WoK	TPA-RAN	TPA-GA	TPA
60	$1.60e+6$ $(5.74e+4)$ ↑	$1.56e+6$ $(1.97e+5)$ ↑	$1.43e+6$ $(5.87e+4)$ ↑	$1.40e+6$ $(2.03e+4)$
80	$2.29e+6$ $(6.89e+4)$ ↑	$2.48e+6$ $(3.14e+5)$ ↑	$2.27e+6$ $(3.14e+4)$ ↑	$2.06e+6$ $(2.68e+4)$
100	$2.99e+6$ $(4.88e+4)$ ↑	$3.63e+6$ $(4.02e+5)$ ↑	$3.18e+6$ $(1.72e+5)$ ↑	$2.68e+6$ $(3.73e+4)$
120	$3.15e+6$ $(5.29e+4)$ ↑	$4.33e+6$ $(1.71e+5)$ ↑	$3.71e+6$ $(1.96e+5)$ ↑	$2.82e+6$ $(6.29e+4)$
140	$4.06e+6$ $(4.88e+4)$ ↑	$5.51e+6$ $(2.29e+5)$ ↑	$4.90e+6$ $(1.24e+5)$ ↑	$3.71e+6$ $(3.03e+4)$
160	$4.66e+6$ $(6.71e+4)$ ↑	$6.61e+6$ $(1.83e+5)$ ↑	$5.94e+6$ $(1.05e+5)$ ↑	$4.21e+6$ $(5.21e+4)$
180	$5.22e+6$ $(6.71e+4)$ ↑	$7.60e+6$ $(1.45e+5)$ ↑	$6.69e+6$ $(1.07e+5)$ ↑	$4.83e+6$ $(4.17e+4)$
200	$5.85e+6$ $(8.88e+4)$ ↑	$8.51e+6$ $(2.38e+5)$ ↑	$7.72e+6$ $(2.62e+5)$ ↑	$5.35e+6$ $(4.20e+4)$
↑ / ↓ / ≈	7/0/0	7/0/0	7/0/0	

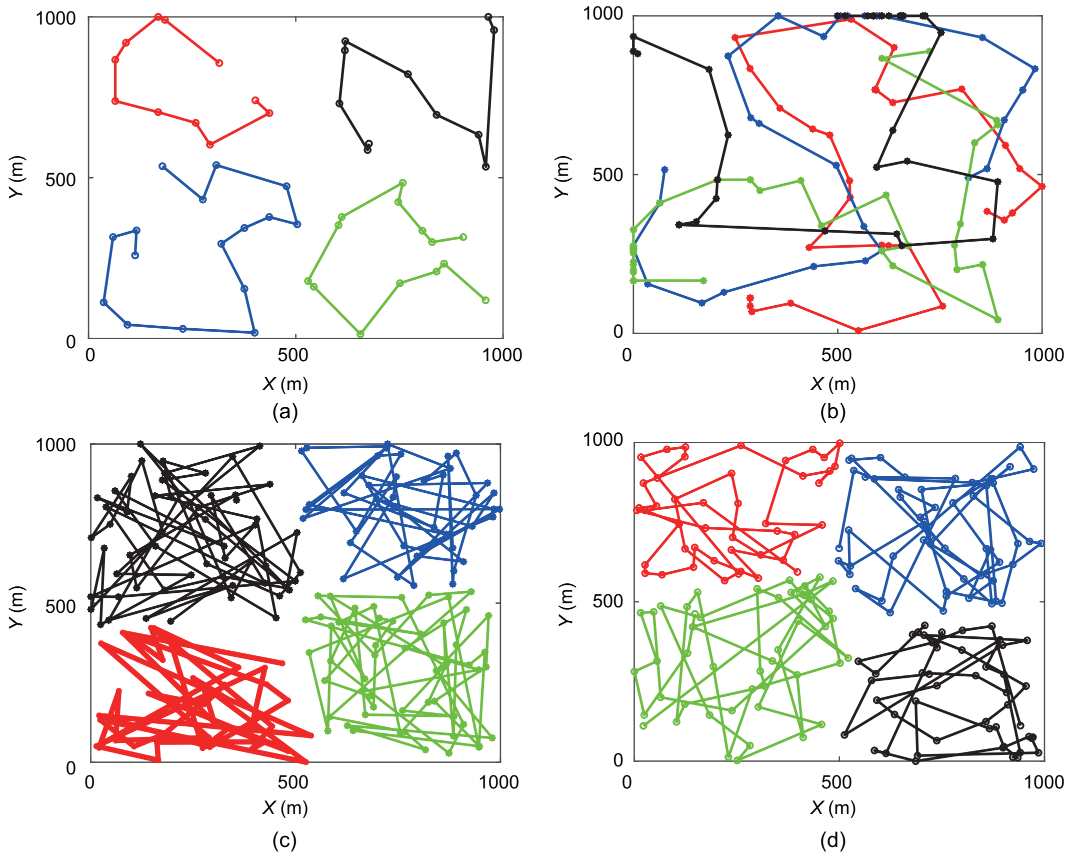

Fig. 6 Trajectories of UAVs obtained by TPA (a), TPA-WoK (b), TPA-RAN (c), and TPA-GA (d) when $n = 200$

Red, green, blue, and black lines indicate trajectories $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, and $\mathcal{G}_4$, respectively. References to color refer to the online version of this figure fact that GA usually requires more fitness evaluations than the greedy method. As a result, under the given time budget, TPA-GA is likely to perform worse than TPA.

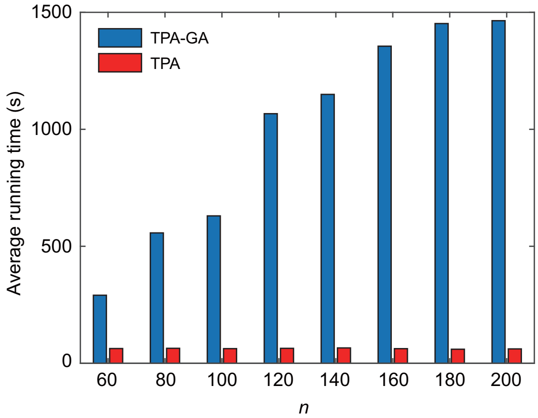

Fig. 7 Average running time of TPA-GA and TPA on each instance

References to color refer to the online version of this figure

4.4 Discussions

4.4.1 Effect of the initial population size

In this study, we set the initial population size of TPA to $k_{\max}$. One might be interested in the effect of the initial population size on the performance of TPA. Therefore, in this subsection, the initial population size is set to a random number in $[k_{\min}, k_{\max}]$. The resulting variant is named TPA-RPS. As shown in Table 3, there is no significant performance difference between TPA and TPA-RPS, which means that the performance of TPA is not sensitive to the initial population size. The reason is that TPA can update the population size adaptively.

Table 3 Experimental results of TPA-RPS and TPA in terms of average EC (J) over 20 runs

n	Mean (standard deviation)	
	TPA-RPS	TPA
60	1.41e + 6 (1.73e + 4) ≈	1.40e + 6 (2.03e + 4)
80	2.06e + 6 (3.19e + 4) ≈	2.06e + 6 (2.68e + 4)
100	2.68e + 6 (5.29e + 4) ≈	2.68e + 6 (3.73e + 4)
120	2.81e + 6 (3.17e + 4) ≈	2.82e + 6 (6.29e + 4)
140	3.70e + 6 (4.06e + 4) ≈	3.71e + 6 (3.03e + 4)
160	4.24e + 6 (5.27e + 4) ≈	4.21e + 6 (5.21e + 4)
180	4.84e + 6 (5.37e + 4) ≈	4.83e + 6 (4.17e + 4)
200	5.36e + 6 (5.59e + 4) ≈	5.35e + 6 (4.20e + 4)
↑ / ↓ / ≈	0/0/7	

4.4.2 Effect of the updating strategy

To investigate the effect of the updating strategy, we test TPA with three different strategies. Specifically, in each updating, at most one, three, and five individuals in the three new populations are different from $\mathcal{P}$. As shown in Table 4, the strategy that can update at most one individual in each updating can provide the best performance among the three compared strategies. The above comparison shows that a dramatic change in population size may lead to a poor performance. Therefore, TPA updates at most one individual in each update.

5 Conclusions

In this study, a multi-UAV-assisted mobile edge computing system was studied. To reduce the energy consumption of the system, a trajectory planning problem was formulated, containing three coupled sub-problems: the deployment of stop points (SPs), the association between UAVs and SPs, and the order of SPs. To solve the trajectory planning problem, we proposed a three-phase trajectory planning algorithm, called TPA. First, differential evolution with a variable population was used for the deployment of SPs, which can simultaneously update the number and locations of SPs. Subsequently, the k-means clustering algorithm was employed to cluster SPs into a set of subsets with the aim of associating the closely spaced SPs with the same UAV. Moreover, to reduce the flight distances of UAVs, we designed a greedy method that can quickly construct the order of SPs visited by UAVs. The experimental results showed that on a set of instances at different scales, TPA can save much energy compared with other algorithms. Therefore, TPA can achieve energy-efficient trajectory planning. However, we need to preset the number of clusters (i.e., the number of UAVs) in the k-means clustering algorithm. As a result, TPA cannot solve the trajectory planning problem for a mobile edge computing system assisted by a variable number of UAVs. In the future, we will try to use the clustering algorithm that does not require a preset number of clusters to solve such a trajectory planning problem.

Contributors

Pei-qiu HUANG and Ke-zhi WANG conceived the idea

Table 4 Experimental results of TPA with three different updating strategies

n	Mean (standard deviation)		
	Five individuals	Three individuals	One individual
60	$1.52e+6$ $(4.74e+4)$ ↑	$1.50e+6$ $(3.63e+4)$ ↑	$1.40e+6$ $(2.03e+4)$
80	$2.22e+6$ $(3.74e+4)$ ↑	$2.15e+6$ $(4.62e+4)$ ↑	$2.06e+6$ $(2.68e+4)$
100	$2.84e+6$ $(7.20e+4)$ ↑	$2.78e+6$ $(5.87e+4)$ ↑	$2.68e+6$ $(3.73e+4)$
120	$3.01e+6$ $(6.73e+4)$ ↑	$2.92e+6$ $(7.72e+4)$ ↑	$2.82e+6$ $(6.29e+4)$
140	$3.95e+6$ $(9.34e+4)$ ↑	$3.81e+6$ $(7.16e+4)$ ↑	$3.71e+6$ $(3.03e+4)$
160	$4.44e+6$ $(1.03e+5)$ ↑	$4.42e+6$ $(8.35e+4)$ ↑	$4.21e+6$ $(5.21e+4)$
180	$5.08e+6$ $(1.10e+5)$ ↑	$4.95e+6$ $(1.05e+5)$ ↑	$4.83e+6$ $(4.17e+4)$
200	$5.75e+6$ $(1.52e+5)$ ↑	$5.53e+6$ $(1.36e+5)$ ↑	$5.35e+6$ $(4.20e+4)$
↑ / ↓ / ≈	7/0/0	7/0/0	

of this study. Yong WANG guided the research and refined the idea. Pei-qiu HUANG performed the research and drafted the manuscript. Ke-zhi WANG discussed the results. Pei-qiu HUANG and Yong WANG revised and finalized the paper.

Compliance with ethics guidelines

Pei-qiu HUANG, Yong WANG, and Ke-zhi WANG declare that they have no conflict of interest.

References

Chan TM, Man KF, Tang KS, et al., 2007. A jumping-genes paradigm for optimizing factory WLAN network. *IEEE Trans Ind Inform*, 3(1):33-43. https://doi.org/10.1109/TII.2006.890528

Chen J, Zhang X, Xin B, et al., 2016. Coordination between unmanned aerial and ground vehicles: a taxonomy and optimization perspective. *IEEE Trans Cybern*, 46(4):959-972. https://doi.org/10.1109/TCYB.2015.2418337

Chen WH, Liu BC, Huang HW, et al., 2019. When UAV swarm meets edge-cloud computing: the QoS perspective. *IEEE Netw*, 33(2):36-43. https://doi.org/10.1109/MNET.2019.1800222

Diao XH, Zheng JC, Cai YM, et al., 2019. Fair data allocation and trajectory optimization for UAV-assisted mobile edge computing. *IEEE Commun Lett*, 23(12):2357-2361. https://doi.org/10.1109/LCOMM.2019.2943461

Du Y, Yang K, Wang KZ, et al., 2019. Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IoT systems. *IEEE Trans Veh Technol*, 68(10):10187-10200. https://doi.org/10.1109/TVT.2019.2935877

Garg S, Singh A, Batra S, et al., 2018. UAV-empowered edge computing environment for cyber-threat detection in smart vehicles. *IEEE Netw*, 32(3):42-51. https://doi.org/10.1109/MNET.2018.1700286

Hu XY, Wong KK, Yang K, et al., 2019. UAV-assisted relaying and edge computing: scheduling and trajectory optimization. *IEEE Trans Wirel Commun*, 18(10):4738-4752. https://doi.org/10.1109/TWC.2019.2928539

Huang L, Wang GC, Bai T, et al., 2017. An improved fruit fly optimization algorithm for solving traveling salesman problem. *Front Inform Technol Electron Eng*, 18(10):1525-1533. https://doi.org/10.1631/FITEE.1601364

Huang PQ, Wang Y, Wang KZ, et al., 2020a. A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. *IEEE Trans Cybern*, 50(10):4228-4241. https://doi.org/10.1109/TCYB.2019.2916728

Huang PQ, Wang Y, Wang KZ, et al., 2020b. Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system. *IEEE Trans Emerg Top Comput Intell*, 4(3):324-335. https://doi.org/10.1109/TETCI.2019.2939373

Jain AK, 2010. Data clustering: 50 years beyond K-means. *Patt Recogn Lett*, 31(8):651-666. https://doi.org/10.1016/j.patrec.2009.09.011

Jeong S, Simeone O, Kang J, 2018. Mobile edge computing via a UAV-mounted cloudlet: optimization of bit allocation and path planning. *IEEE Trans Veh Technol*, 67(3):2049-2063. https://doi.org/10.1109/TVT.2017.2706308

Jin MS, Gao S, Luo HB, et al., 2019. Cost-effective resource segmentation in hierarchical mobile edge clouds. *Front Inform Technol Electron Eng*, 20(9):1209-1220. https://doi.org/10.1631/FITEE.1800203

Li MS, Cheng N, Gao J, et al., 2020. Energy-efficient UAV-assisted mobile edge computing: resource allocation and trajectory optimization. *IEEE Trans Veh Technol*, 69(3):3424-3438. https://doi.org/10.1109/TVT.2020.2968343

Low JE, Sufiyan D, Win LST, et al., 2019. Design of a hybrid aerial robot with multi-mode structural efficiency and optimized mid-air transition. *Unmann Syst*, 7(4):195-213. https://doi.org/10.1142/S2301385019500067

Mozaffari M, Saad W, Bennis M, et al., 2019. A tutorial on UAVs for wireless networks: applications, challenges, and open problems. *IEEE Commun Surv Tutor*, 21(3):2334-2360. https://doi.org/10.1109/COMST.2019.2902862

Ryerkerk M, Averill R, Deb K, et al., 2019. A survey of evolutionary algorithms using metameric representations. *Genet Program Evol Mach*, 20(4):441-478. https://doi.org/10.1007/s10710-019-09356-2

Storn R, Price K, 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim*, 11(4):341-359. https://doi.org/10.1023/A:1008202821328

Ting CK, Lee CN, Chang HC, et al., 2009. Wireless heterogeneous transmitter placement using multiobjective variable-length genetic algorithm. *IEEE Trans Syst Man Cybern Part B (Cybern)*, 39(4):945-958. https://doi.org/10.1109/TSMCB.2008.2010951

Wang BC, Li HX, Zhang QF, et al., 2018. Decomposition-based multiobjective optimization for constrained evolutionary optimization. *IEEE Trans Syst Man Cybern Syst*, in press. https://doi.org/10.1109/TSMC.2018.2876335

Wang JH, Zhou Y, Wang Y, et al., 2016. Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms. *IEEE Trans Cybern*, 46(3):582-594. https://doi.org/10.1109/TCYB.2015.2409837

Wang KZ, Huang PQ, Yang K, et al., 2019. Unified offloading decision making and resource allocation in ME-RAN. *IEEE Trans Veh Technol*, 68(8):8159-8172. https://doi.org/10.1109/TVT.2019.2926513

Wang L, Huang PQ, Wang KZ, et al., 2019. RL-based user association and resource allocation for multi-UAV enabled MEC. Proc 15th Int Wireless Communications & Mobile Computing Conf, p.741-746. https://doi.org/10.1109/IWCMC.2019.8766458

Wang Y, Cai ZX, Zhang QF, 2011. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans Evol Comput*, 15(1):55-66. https://doi.org/10.1109/TEVC.2010.2087271

Wang Y, Liu H, Long H, et al., 2018. Differential evolution with a new encoding mechanism for optimizing wind farm layout. *IEEE Trans Ind Inform*, 14(3):1040-1054. https://doi.org/10.1109/TII.2017.2743761

Wang Y, Ru ZY, Wang KZ, et al., 2020. Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. *IEEE Trans Cybern*, 50(9):3984-3997. https://doi.org/10.1109/TCYB.2019.2935466

Xin B, Chen J, Zhang J, et al., 2012. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. *IEEE Trans Syst Man Cybern Part C (Appl Rev)*, 42(5):744-767. https://doi.org/10.1109/TSMCC.2011.2160941

Xu JW, Ota K, Dong MX, et al., 2018. SIoTFog: Byzantine-resilient IoT fog networking. *Front Inform Technol Electron Eng*, 19(12):1546-1557. https://doi.org/10.1631/FITEE.1800519

Yang ZH, Pan CH, Wang KZ, et al., 2019. Energy efficient resource allocation in UAV-enabled mobile edge computing networks. *IEEE Trans Wirel Commun*, 18(9):4576-4589. https://doi.org/10.1109/TWC.2019.2927313

Zaini AH, Xie LH, 2020. Distributed drone traffic coordination using triggered communication. *Unmann Syst*, 8(1):1-20. https://doi.org/10.1142/S2301385020500016

Zhang J, Huang T, Wang S, et al., 2019. Future Internet: trends and challenges. *Front Inform Technol Electron Eng*, 20(9):1185-1194. https://doi.org/10.1631/FITEE.1800445

Zhang J, Zhou L, Zhou FH, et al., 2020. Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing. *IEEE Trans Veh Technol*, 69(2):2114-2125. https://doi.org/10.1109/TVT.2019.2960103

Zhang L, Zhao Z, Wu QW, et al., 2018. Energy-aware dynamic resource allocation in UAV assisted mobile edge computing over social Internet of vehicles. *IEEE Access*, 6:56700-56715. https://doi.org/10.1109/ACCESS.2018.2872753

Zollars MD, Cobb RG, Grymin DJ, 2019. Optimal SUAS path planning in three-dimensional constrained environments. *Unmann Syst*, 7(2):105-118. https://doi.org/10.1142/S2301385019500031