Joint Optimization of Deployment and Flight Planning of Multi-UAVs for Long-distance Data Collection from Large-scale IoT Devices

Yiying Zhang, Yue Huang, Chao Huang, Hailong Huang, Tran Anh-Tu Nguyen

Joint Optimization of Deployment and Flight Planning of Multi-UAVs for Long-distance Data Collection from Large-scale IoT Devices

Yiying Zhang, Yue Huang, Chao Huang, Hailong Huang, and Anh-Tu Nguyen

***Abstract*—Internet of Things (IoT) devices have been widely deployed to build smart cities. How to efficiently collect data from large-scale IoT devices is a valuable and challenging research topic. Benefiting from agility, flexibility, and deployability, an unmanned aerial vehicle (UAV) has great potential to be an aerial base station. However, given the limited battery capacity, the flight time of a UAV is limited. This paper focuses on using multi-UAVs to execute long-distance data collection from large-scale IoT devices. We design a multi-UAVs-assisted large-scale IoT data collection system. The core facilities of this system are the data center and charging stations, which are equipped with a limited number of charging piles to provide charging services for UAVs. To ensure the efficient operation of the system, the problem of deployment and flight planning of UAVs is formulated as a joint optimization problem. To solve the problem, a population-based optimization algorithm with a three-layer structure, namely EDDE-DPDE, is proposed. It includes two core components: elite-driven differential evolution (EDDE) and differential evolution with a dynamic population (DPDE), which are two variants of differential evolution. Thanks to ideas of reusing elite individuals and historical information, the proposed EDDE-DPDE shows an improvement of at least 11.11% compared with four powerful algorithms in terms of average travel time.**

***Index Terms*—Data collection, Internet of Things, multi-UAVs, flight planning, differential evolution.**

I. Introduction

In recent years, the concept of the smart city is put forward to provide a comfortable life to citizens by fully utilizing modern technology, such as super-computing systems, Internet of Things (IoT), big data analysis, and unmanned aerial vehicles (UAVs). The major function of IoT in building smart cities is to collect and send data with the assistance of internet-connected devices [1]. For some stand-alone IoT devices, their energy supply is constrained. Thus, the efficient usage of limited energy is of great concern.

This work was supported by Department General Research Grant [P0040253] and funding from the Research Institute for Sports Science and Technology [P0043566]. *(Corresponding author: Chao Huang.)*

Y. Zhang and C. Huang are with the Department of Industrial and Systems Engineering, the Hong Kong Polytechnic University, Hong Kong. (E-mail: {yiyingchreo.zhang, hchao.huang}@polyu.edu.hk).

Y. Huang is with Data61 Commonwealth Scientific and Industrial Research Organisation, Melbourne, Australia (Email: yuehuangcsiro@gmail.com).

H. Huang is with the Department of Aeronautical and Aviation Engineering and The Research Institute for Research Institute for Sports Science and Technology (RISports), the Hong Kong Polytechnic University, Hong Kong. (E-mail: hailong.huang@polyu.edu.hk).

A. Nguyen is with the LAMIH laboratory, UMR CNRS 8201, Universit Polytechnique Hauts-de-France, 59300 Valenciennes, France, and also with the INSA Hauts-de-France, 59300 Valenciennes, France (e-mail: nguyen.trananhtu@gmail.com).

UAVs can play many roles in building smart cities, such as target monitoring, logistics distribution, topographic mapping, and power line inspection [2]–[5]. In addition, a UAV can also be regarded as an aerial base station to collect data from IoT devices [6]–[8]. A lot of effort has been conducted for improving the efficiency of using UAVs to collect data from IoT devices [9]–[21]. However, they do not cover how to use multi-UAVs simultaneously to carry out long-distance data collection tasks for large-scale IoT devices. There are two challenges associated with this topic:

- With the continuous advancement of smart city construction, it is a trend that large-scale IoT devices are deployed in all aspects of the city. It is a promising approach to deploy multi-UAVs to improve collection efficiency. Specifically, multiple stop points are required for a UAV to collect data from large-scale IoT devices. How to determine the optimal number and locations of stop points for a UAV is a challenge.
- In a real collection scenario, a UAV needs to fly from a starting point to a destination point. Then, it executes the collection task at the destination point, after which, it returns to the starting point. Then, how to support long-distance flights and design efficient flight planning (FP) for multi-UAVs is another challenge.

Motivated by these challenges, this paper aims at solving the joint optimization problem of deployment and FP for a multi-UAVs-assisted large-scale IoT data collection system. This problem is to minimize the average travel time of all UAVs to complete data collection. To solve the problem, a new population-based optimization algorithm named EDDE-DPDE is proposed. The main contributions are as follows:

- This paper builds a multi-UAVs-assisted large-scale IoT data collection system. It contains four core components: a data center (DC), some UAVs, large-scale IoT devices, and some charging stations (CSs). The DC performs large-scale data processing tasks and provides daily maintenance and storage for UAVs. The DC and CSs are equipped with a limited number of charging piles, which form a charging service network (CSN) to support multi-UAVs to execute long-distance data collection.
- A new population-based optimization algorithm, namely EDDE-DPDE, is proposed to solve the above problem. Two core components of EDDE-DPDE are elite-driven differential evolution (EDDE) and differential evolution with a dynamic population (DPDE). EDDE-DPDE has

TABLE I: The summary of some key references.

Reference	Deployment optimization	Flight planning
[9]	×	✓
[10]	✓	×
[11]	×	✓
[12]	×	✓
[13]	✓	×
[14]	✓	×
[15]	×	✓
[16]	×	✓
[17]	×	✓
[18]	×	✓
[19]	×	✓
[20]	✓	✓
[21]	✓	✓

a three-layer structure. EDDE is first used to search the optimal FP that all UAVs fly from the DC to their destination charging stations (DCSs). Then, DPDE is employed to determine the optimal deployment of all UAVs during data collection. Lastly, EDDE is applied again to search the optimal FP that all UAVs fly from their DCSs to the DC.

- This paper designs a new variant of differential evolution (DE), namely EDDE, to determine the optimal FP for all UAVs. The FP problem is converted into a large-scale optimization problem with the help of the priority sequence (PS) generated by a priority-based encoding mechanism. The basic idea of EDDE is to enhance the global search ability of DE by making full use of the obtained best FP, which is reflected in three aspects: 1) an elite archive used to save some promising individuals is created, and a new mutation operator is designed by reusing these promising individuals, 2) a new crossover operator is introduced based on the obtained best PF, and 3) a local escape operator with random numbers produced by opposition-based learning is presented, which is a further optimization for the obtained best FP.
- This paper presents another variant of DE, namely DPDE, to determine the optimal deployment of all UAVs during data collection. The basic idea of DPDE is to adaptively search the optimal number and locations of stop points of UAVs, which is reflected in two aspects: 1) three mutation strategies are designed by reusing historical population information, and 2) a random crossover operator is designed based on random numbers with a uniform distribution between 0 and 1.
- This paper simulates a realistic scenario of using multi-UAVs to execute long-distance data collection tasks with the multi-UAVs-assisted large-scale IoT data collection system. Specifically, the nearest center rule is made to assign the data collection task and determine the DCS for each UAV. The impact of EDDE and DPDE on the performance of EDDE-DPDE is investigated by comparing with four powerful algorithms.

The rest of this paper includes five sections. Related work is introduced in Section II. Section III and Section IV present the system model and problem formulation, respectively. Section V describes the proposed algorithms. Experimental results are discussed in Section VI. Section VII concludes this paper.

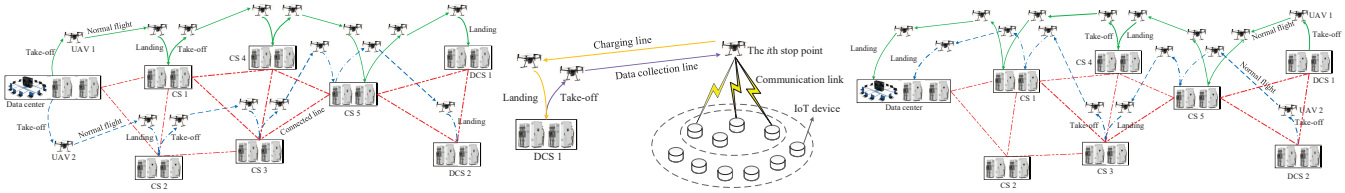
II. Related work

In recent years, using the UAV to collect data from IoT devices has attracted the interest of many scholars as shown in Table I. Wang et al. [9] propose a novel UAV-assisted IoT network to provide energy-efficient data collection service. Huang et al. [10] design a differential evolution with a variable population size to optimize the deployment of a UAV in a UAV-assisted IoT data collection system. Samir et al. [11] use a UAV to collect data from time-constrained IoT devices. Hu et al. [12] investigate the UAV-assisted wireless-powered IoT system. Li et al. [13] employ an energy-constrained UAV to collect data of IoT devices in a sparse IoT-sensor network. Ghdiri et al. [14] adopt UAVs to collect data from time-constrained sensor nodes. Liu et al. [15] aim at considering a scenario of using UAVs to collect data from a dynamic IoT network. Wang et al. [16] pay attention to employing UAVs to collect data from distributed IoT nodes. In [17], a UAV-assisted multicarrier wireless-powered communication model is proposed for IoT scenarios. In [18], resource allocation and trajectory design for UAV-assisted full-duplex IoT networks with the emergency communication system are investigated. In [19], a UAV-supported clustered non-orthogonal multiple access system is designed for 6G-enabled IoT. The mentioned references focus on improving the efficiency of data collection by optimizing the deployment or FP of UAVs and do not address the FP and deployment of UAVs at the same time.

In addition, although the FP and deployment problems of UAVs are addressed simultaneously in [20], [21], they do not cover large-scale IoT devices and long-distance flights of UAVs. Specifically, the considered deployment and FP problems have their characteristics:

- The deployment problem. Considering the large workload, multi-UAVs are employed. In addition, a UAV needs to collect data at multiple stop points and add its energy multiple times due to the heavy workload. Thus, the considered deployment problem is to determine the optimal number and locations of stop points for multi-UAVs, which takes the least average travel time with the guarantee of adding energy for multi-UAVs.
- The FP problem. The considered FP problem is to determine the optimal FP for multi-UAVs that can take the least average travel time on the round trip paths between the DC and their DCSs. Note that, the distance between the DC and their DCSs is usually out of their maximum range. Thus, how to simultaneously support multi-UAVs to carry out long-distance flights is the main challenge of the considered FP problem.

The deployment and FP problems of UAVs can be regarded as optimization problems. The superiority of population-based metaheuristic algorithms on such problems has been proven [9], [10], [21]–[23]. The basic idea of the population-based metaheuristic algorithm is as follows. Each individual in the population represents a solution to the solved problem. All individuals share the obtained valuable information, which generate the next generations by the defined iteration rules inspired by natural phenomenon. The iteration rules have two remarkable characteristics: 1) they consider both local



(a) Stage 1: UAVs fly from the DC to their DCSs. (b) Stage 2: a UAV executes data collection. (c) Stage 3: UAVs fly from their DCSs to the DC.

Fig. 1. The operation of the built multi-UAVs-assisted large-scale IoT data collection system ("connected line" is that the connected two facilities are reachable; "charging line" is that a UAV returns to its DCS to recharge; "data collection line" is that a UAV returns to the operating point for data collection).

search and global search, and 2) they refer to random numbers that follow uniform distribution or normal distribution. Thus, population-based metaheuristic algorithms have more chances to find better solutions than numerical algorithms in solving complex engineering problems [24].

At present, two main methods support the long-distance flight of a UAV, which are the UAV-vehicle model and the UAV-station model [25]. UAV-vehicle model is the cooperation between the UAV and the vehicle. The vehicle can add the energy of a UAV by replacing the battery or providing charging service [26]. UAV-station model can add the energy of a UAV by the deployed charging stations on the ground [25]. Given the uncertainty of the traffic, the UAV-station model outperforms the UAV-vehicle model in terms of reliability.

Motivated by the mentioned research, this paper builds a multi-UAVs-assisted large-scale IoT data collection system based on the UAV-station model and designs a new population-based metaheuristic algorithm based on the characteristics of the considered deployment and FP problems.

III. System Model

On the premise of being close to a real scenario, a multi-UAVs-assisted large-scale IoT data collection system is built as shown in Fig. 1. It consists of a DC, some CSs, some UAVs, and large-scale IoT devices. The DC plays two roles: 1) processing the collected data, and 2) providing some services to a UAV, such as charging and maintenance. The DC and CSs are equipped with a limited number of charging piles to provide charging services to UAVs, which are called nodes that form a CSN to support long-distance flights of multi-UAVs. A UAV is a platform for collecting data from IoT devices, which has three working stages: 1) it flies from the DC to its operating area with the help of CSN as shown in Fig. 1(a), 2) it executes collection task at its operating area as shown in Fig. 1(b), and 3) it returns to the DC with the help of CSN after completing its task as shown in Fig. 1(c).

We take UAV 1 in Fig. 1 as an example to describe the operation of this system. In the first stage, UAV 1 flies from the DC to DCS 1 with the help of CSN. DCS 1 is the closest CS to the operating area of UAV 1 among all CSs. In the second stage, UAV 1 executes the data collection task. During data collection, UAV 1 can return to DCS 1 for charging. In the third stage, UAV 1 flies from DCS 1 to the DC. Note that, given the limited CSs, UAV 1 needs to compete with other UAVs for charging resources, such as CS 5 in Fig. 1(a) and CS 1 and CS 5 in Fig. 1(c). Clearly, as the number of UAVs increases, UAV 1 will bear greater competitive pressure, which also applies to other UAVs.

IV. Problem Statement

This section first introduces preliminary knowledge. Then, the considered deployment problem and FP problem are stated. In addition, to convert the considered practical problems into mathematical models, the following assumptions are made:

- All UAVs are identical, are fully charged at the DC and do not collide with obstacles during the task.
- A UAV has three flight states: take-off, landing, and normal flight. Specifically, a UAV flies between two nodes in a straight line with a constant speed and altitude, which takes off and lands vertically above the take-off point and landing point, respectively.
- The factors on the energy consumption of a UAV involve take-off, landing, charging, normal flight, and data collection.
- The factors on the travel time of a UAV involve take-off, landing, normal flight, charging, wait, and data collection.
- After a UAV establishes a communication link with an IoT device, the communication link will not be disturbed by other factors.

Remark 1. *This work focuses on the high-level flight planning problem of multiple UAVs rather than the low-level motion control problem. So, this work does not present the UAV kinematics and dynamics models. In the considered context, to conduct a mission, a UAV normally takes off, does a level flight to the destination and then lands. During these processes, classic algorithms such as PID can easily achieve the expected motion control.*

A. Preliminary knowledge

This section introduces the charging function, distance relationships between nodes and status chart of charging piles.

1) Charging function: A node in the CSN can provide charging service to UAVs. A UAV is typically equipped with series-connected lithium-ion polymer battery cells [27], whose charging process follows a "slow-fast-slow" curve [28]. Function (1) describes the charging process:

$$C = \frac{C_0}{1 + \exp(6 - t/5)}, \tag{1}$$

where C_0 is the maximum energy capacity of a battery, C is the residual energy of a battery, and t is charging time.

Algorithm 1 Calculate $T_{i,j}$ and $t_{\mathrm{w},k,i}$

Input:
$T_{i,j}$, $t_{\mathrm{v},k,i}$, $t_{\mathrm{w},k,i}$, $t_{\mathrm{w},k,i,j}$
Output:
$t_{\mathrm{w},k,i}$, $T_{i,j}$
Initialize flag bit f_{t} by $f_{\mathrm{t}}=0$ and $t_{\mathrm{w},k,i,j}$ by $t_{\mathrm{w},k,i,j}=0$;
for $j=1$ to n_{c} **do**
  if $t_{\mathrm{v},k,i} > t_{\mathrm{e},i,j}$ **then**
    $f_{\mathrm{t}}=1$, $t_{\mathrm{w},k,i}=0$, $t_{\mathrm{s},i,j}$=$t_{\mathrm{v},k,i}$, $t_{\mathrm{e},i,j}=t_{\mathrm{v},k,i}+t_{\mathrm{c},k,i}$;
    Break;
  else
    $t_{\mathrm{w},k,i,j}=t_{\mathrm{e},i,j}-t_{\mathrm{v},k,i}$;
  end if
end for
if $f_{\mathrm{t}}==0$ **then**
  Search the charging pile j^* corresponding to the minimum wait time;
  $t_{\mathrm{w},k,i}=t_{\mathrm{w},k,i,m}$, $t_{\mathrm{s},i,j^*}=t_{\mathrm{v},k,i}$, $t_{\mathrm{e},i,j^*}=t_{\mathrm{v},k,i}+t_{\mathrm{c},k,i}+t_{\mathrm{w},k,i}$.
end if

2) Distance relationships between nodes: When the distance between two nodes in the CSN is beyond the maximum range of a UAV, the UAV cannot fly between the two nodes. Let n_{d}, n_{s}, and S denote the number of DC, the number of CS, and the sequence of numbers of all nodes, respectively. S can be expressed by:

$$S=[1,2,\ldots,n_{\mathrm{d}},n_{\mathrm{d}}+1,n_{\mathrm{d}}+2,\ldots,n_{\mathrm{d}}+n_{\mathrm{s}}]. \tag{2}$$

Let n_{n} and $D_{\max}$ denote the total number of nodes in the CSN and the maximum range of a UAV, respectively. If the distance between two nodes is less than $D_{\max}$, the two nodes are connected; otherwise, the two nodes are not connected. From (2), the distance relationship among nodes can be expressed by a 0-1 matrix with n rows and n columns, which is called the connected matrix $\boldsymbol{M}_{\mathrm{CSN}}$. Specifically, in $\boldsymbol{M}_{\mathrm{CSN}}$, "0" denotes that it is not reachable; "1" denotes that it is reachable.

3) Status chart of charging piles: Let n_{c} denote the number of charging piles at one node. When the number of UAVs (which need to be charged) is larger than n_{c}, some UAVs need to wait until idle charging piles appear. This paper uses a status chart to record the status of charging piles, which can be described by:

$$T_{i,j}=[t_{\mathrm{s},i,j}\ t_{\mathrm{e},i,j}], i\in[1,n_{\mathrm{n}}], j\in[1,n_{\mathrm{c}}], \tag{3}$$

where $T_{i,j}$, $t_{\mathrm{s},i,j}$, and $t_{\mathrm{e},i,j}$ are the occupied period, start time, and end time of charging pile j at node i, respectively. Let $t_{\mathrm{v},k,i}$, $t_{\mathrm{w},k,i}$, and $t_{\mathrm{w},k,i,j}$ denote the time of UAV k arriving at node i, the wait time of UAV k at node i, and the wait time of UAV k at the jth charging pile of node i, respectively. $T_{i,j}$ and $t_{\mathrm{w},k,i}$ can be computed by Algorithm 1. Here, $k\in[1,n_{\mathrm{u}}]$ and n_{u} is the number of UAVs.

B. The FP problem

The considered FP of a UAV includes: 1) the UAV flies from the DC to its DCS as shown in Fig. 1(a), and 2) the UAV flies from its DCS to the DC as shown in Fig. 1(c). According to Fig. 1, when a UAV flies from one node to another, its FP can be seen as a sequence consisting of some nodes. Let $\boldsymbol{L}_{\mathrm{a}}=\{\boldsymbol{L}_{\mathrm{a},1},\boldsymbol{L}_{\mathrm{a},2},\ldots,\boldsymbol{L}_{\mathrm{a},n_{\mathrm{u}}}\}$ and $\boldsymbol{L}_{\mathrm{c}}=\{\boldsymbol{L}_{\mathrm{c},1},\boldsymbol{L}_{\mathrm{c},2},\ldots,\boldsymbol{L}_{\mathrm{c},n_{\mathrm{u}}}\}$ denote the flights of all UAVs in stage 1 and stage 3, respectively. Specifically, $\boldsymbol{L}_{\mathrm{a},k}$ and $\boldsymbol{L}_{\mathrm{c},k}$ are the flight of UAV k in the stage 1 and stage 3, respectively. $\boldsymbol{L}_{\mathrm{a},k}$ and $\boldsymbol{L}_{\mathrm{c},k}$ can be denoted by:

$$\boldsymbol{L}_{\mathrm{a},k}=[L_{\mathrm{a},k,1},L_{\mathrm{a},k,2},\ldots,L_{\mathrm{a},k,l_{\mathrm{a},k}}], l_{\mathrm{a},k}\geq 3, \tag{4}$$

$$\boldsymbol{L}_{\mathrm{c},k}=[L_{\mathrm{c},k,1},L_{\mathrm{c},k,2},\ldots,L_{\mathrm{c},k,l_{\mathrm{c},k}}], l_{\mathrm{c},k}\geq 3, \tag{5}$$

where $l_{\mathrm{a},k}$ and $l_{\mathrm{c},k}$ are the lengths of $\boldsymbol{L}_{\mathrm{a},k}$ and $\boldsymbol{L}_{\mathrm{c},k}$, respectively.

The travel time consumed on $\boldsymbol{L}_{\mathrm{a},k}$ consists of four parts:

- Take-off and landing time $t_{\mathrm{tol},k}$. From (4), UAV k takes off $l_{\mathrm{a},k}-1$ times and lands $l_{\mathrm{a},k}-1$ times. Thus, $t_{\mathrm{tol},k}$ can be obtained by:
$$t_{\mathrm{tol},k}=t_{\mathrm{tl}}(l_{\mathrm{a},k}-1), \tag{6}$$
where t_{tl} is the consumed total time by UAV k on taking off once and landing once.
- Normal flight time $t_{\mathrm{nf},k}$. Let v denote the constant speed of UAV k on the normal flight. $t_{\mathrm{nf},k}$ is computed by:
$$t_{\mathrm{nf},k}=\sum_{s=1}^{L_{\mathrm{a},k,l-1}}\frac{d_{L_{\mathrm{a},k,s},L_{\mathrm{a},k,s+1}}}{v}, \tag{7}$$
where $d_{L_{\mathrm{a},k,s},L_{\mathrm{a},k,s+1}}$ is the Euclidean distance between node $L_{\mathrm{a},k,s}$ and node $L_{\mathrm{a},k,s+1}$.
- Charging time $t_{\mathrm{ct},k}$. Let $t_{k,L_{\mathrm{a},k,h}}$ denote the charging time of UAV k at node $L_{\mathrm{a},k,h}$, where h is an integer between 1 and $l_{\mathrm{a},k}$. Let $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$ and $C_{\mathrm{n},k,L_{\mathrm{a},k,h}}$ are the rest energy and required energy of UAV k at node $L_{\mathrm{a},k,h}$, respectively. From (1), to get $t_{k,L_{\mathrm{a},k,h}}$, $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$ and $C_{\mathrm{n},k,L_{\mathrm{a},k,h}}$ need to be computed. To improve the flight efficiency, $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$ is the minimum energy that can allow UAV k to fly from node $L_{\mathrm{a},k,h}$ to node $L_{\mathrm{a},k,h+1}$. In addition, the penalty energy C_{p} is introduced to prevent a UAV from running out of energy. Thus, $C_{\mathrm{n},k,L_{\mathrm{a},k,h}}$ can be achieved by:
$$C_{\mathrm{p}}=\eta C_0, \tag{8}$$
$$C_{\mathrm{n},k,L_{\mathrm{a},k,h}}=\frac{P_0 d_{L_{\mathrm{a},k,h},L_{\mathrm{a},k,h+1}}}{v}+C_{\mathrm{tl}}+C_{\mathrm{p}}, \tag{9}$$
$$C_{\mathrm{n},k,L_{\mathrm{a},k,h}}=C_0,\ \text{if } C_{\mathrm{n},k,L_{\mathrm{a},k,h}}>C_0, \tag{10}$$
where η is the penalty factor that is set to 0.1 in this paper, C_{tl} is the consumed total energy on take-off once and landing once, and P_0 is the rated power of UAV k. Note that, (10) is to make $C_{\mathrm{n},k,L_{\mathrm{a},k,h}}$ obtained from (9) not more than the maximum energy capacity of a UAV. To compute $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$, the real penalty energy $C_{\mathrm{np},k,L_{\mathrm{a},k,h}}$ is defined by:
$$C_{\mathrm{np},k,L_{\mathrm{a},k,h}}=\begin{cases} C_0-C_{\mathrm{n},k,L_{\mathrm{a},k,h}},\ \text{if } C_{\mathrm{r},k,L_{\mathrm{a},k,h}}>\mathrm{C}_0; \\ C_{\mathrm{p}},\text{if } C_{\mathrm{n},k,L_{\mathrm{a},k,h}}\leq C_0. \end{cases} \tag{11}$$
According to (11), $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$ can be computed by:
$$C_{\mathrm{r},k,L_{\mathrm{a},k,h}}=\begin{cases} C_0-\frac{P_0 d_{L_{\mathrm{a},k,h},L_{\mathrm{a},k,h+1}}}{v}-C_{\mathrm{tl}},\text{if } h=2; \\ C_{\mathrm{np},k,L_{\mathrm{a},k,h}},\text{if } h\in(2,l_{\mathrm{a},k}). \end{cases} \tag{12}$$
Let $t_{\mathrm{ct},k,L_{\mathrm{a},k,h}}$ denote the charging time of UAV k at node $L_{\mathrm{a},k,h}$. $t_{\mathrm{ct},k,L_{\mathrm{a},k,h}}$ can be computed by substituting

$C_{\mathrm{n},k,L_{\mathrm{a},k,h}}$ obtained from (8-10) and $C_{\mathrm{r},k,L_{\mathrm{a},k,h}}$ obtained from (11-12) into (1)), which can be described by:

$$t_{\mathrm{ct},k,L_{\mathrm{a},k,h}} = 5\ln(\frac{C_0}{C_{\mathrm{r},k,L_{\mathrm{a},k,h}}} - 1) - 5\ln(\frac{C_0}{C_{\mathrm{n},k,L_{\mathrm{a},k,h}}} - 1). \tag{13}$$

Thus, $t_{\mathrm{tc},k}$ can be expressed by:

$$t_{\mathrm{ct},k} = \sum_{h=2}^{L_{\mathrm{a},k-1}} t_{\mathrm{ct},k,L_{\mathrm{a},k,h}}. \tag{14}$$

- Wait time $t_{\mathrm{wt},k}$. From Algorithm 1, $t_{\mathrm{wt},k}$ is obtained by:

$$t_{\mathrm{wt},k} = \sum_{h=2}^{L_{\mathrm{a},k-1}} t_{\mathrm{w},k,L_{\mathrm{a},k,h}}. \tag{15}$$

Let $t_{\mathrm{fdc},k}$ and $t_{\mathrm{fcd},k}$ denote the travel time on $\boldsymbol{L}_{\mathrm{a},k}$ and $\boldsymbol{L}_{\mathrm{c},k}$, respectively. Based on (6), (7), (14), and (15), $t_{\mathrm{fdc},k}$ can be computed by:

$$t_{\mathrm{fdc},k} = t_{\mathrm{tl},k} + t_{\mathrm{nf},k} + t_{\mathrm{ct},k} + t_{\mathrm{wt},k}. \tag{16}$$

Similarly, $t_{\mathrm{fcd},k}$ also can be obtained according to (1-16).

C. The deployment problem

As shown in Fig. 1(b), the consumed time on the deployment of UAV k is associated with two aspects:

1) Data collection time: Let $n_{\mathrm{IoT},k}$ denote the number of IoT devices assigned to UAV k and $(x_i, y_i, 0)$ denote the location of IoT device i ($i \in [1, n_{\mathrm{IoT},k}]$), where x_i and y_i are the coordinate values in the x-axis and y-axis of the location of IoT device i, respectively. Let $n_{\mathrm{sp},k}$ denote the number of stop points of UAV k to complete the assigned collection task and $(X_{k,j}, Y_{k,j}, H_{k,j})$ denote the location of stop point j ($j \in [1, n_{\mathrm{sp},k}]$), where $X_{k,i}$ and $Y_{k,i}$, and $H_{k,i}$ are the coordinate values in the x-axis, y-axis, and z-axis of the location of stop point j, respectively.

Like in [10], a binary variable $c_{\mathrm{s},i,j}$ is used to record the connection status between IoT device i and stop point j. If $c_{\mathrm{s},i,j}$ is equal to 0, the connection is closed; otherwise, the connection is open. In addition, IoT device i sends its data to the nearest stop point. So, $c_{\mathrm{s},i,j}$ can be written by:

$$c_{\mathrm{s},i,j} = \begin{cases} 1, \text{ if } j = \underset{j \in 1,2,\ldots,n_{\mathrm{sp},k}}{\arg\min}\, d_{ij}, i \in [1, n_{\mathrm{IoT},k}]; \\ 0, \text{ otherwise.} \end{cases} \tag{17}$$

$$\sum_{j=1}^{n_{\mathrm{sp},k}} c_{\mathrm{s},i,j} = 1, \ i \in [1, n_{\mathrm{IoT},k}], \tag{18}$$

$$\sum_{i=1}^{n_{\mathrm{IoT},k}} \sum_{j=1}^{n_{\mathrm{sp},k}} c_{\mathrm{s},i,j} = n_{\mathrm{IoT},k}, \tag{19}$$

where $d_{i,j}$ is the Euclidean distance between between IoT device i and stop point j. (18) is to ensure that IoT device i only sends data to UAV k once. (19) is to ensure that all IoT devices can send data to UAV k. In addition, considering the bandwidth, the number of IoT devices simultaneously sending data to UAV k does not exceed n_{sd}, which can be denoted by:

$$\sum_{i=1}^{n_{\mathrm{IoT},k}} c_{\mathrm{s},i,j} \le n_{\mathrm{sd}}, \ j \in [1, n_{\mathrm{sp},k}]. \tag{20}$$

The channel modeling is same with that of [10]. In this channel modeling, an IoT device sends its data to a UAV by a wireless channel. The data rate $R_{i,j}$ can be computed by [10]:

$$R_{ij} = B\log_2^{1+\frac{P_i G_{ij}}{\delta^2}} = B\log_2^{1+\frac{P_i G_0 d_{ij}^{-2}}{\delta^2}}, \tag{21}$$

where B is the system bandwidth, δ^2 is the white Gaussian noise power, $G_{i,j}$ is the channel gain between UAV k at stop point j and IoT device i, G_0 is the channel power gain at the reference distance $d_0 = 1$m, and P_i is the transmitting power between IoT device i and UAV k. Thus, when IoT device i sends D_i amount of data to UAV k at the stop point j, the required transmission time $t_{\mathrm{tt},i}$ can be obtained by:

$$t_{\mathrm{tt},i} = \frac{D_i}{R_{ij}}. \tag{22}$$

Let $t_{\mathrm{ht},k,j}$ and $C_{\mathrm{ht},k,j}$ denote the hover time and consumed energy of UAV k at stop point j, respectively. According to (20), $t_{\mathrm{ht},k,j}$ and $C_{\mathrm{ht},k,j}$ can be represented by:

$$t_{\mathrm{ht},k,j} = \max(t_{\mathrm{tt},i}), i \in [1, n_{\mathrm{IoT},k}], \tag{23}$$

$$C_{\mathrm{ht},k,j} = t_{\mathrm{ht},k,j} P_{\mathrm{hp}}, \tag{24}$$

where P_{hp} is the hover power of UAV k.

2) Round-trip time: Looking at Fig. 1(b), UAV k needs to return to its DCS to charge its battery during data collection. Let $D_{\mathrm{cs},k}$, $\boldsymbol{L}_{\mathrm{sp},k}$, $\boldsymbol{T}_{\mathrm{ht},k}$, $\boldsymbol{C}_{\mathrm{he},k}$, and $t_{\mathrm{ctt},k}$ denote the location of the DCS of UAV k, the set of locations of all stop points of UAV k, the set of of the consumed hover time at all stop points of UAV k, the set of of the consumed hover energy at all stop points of UAV k, and the travel time of UAV k during data collection, respectively. Specifically, $\boldsymbol{L}_{\mathrm{sp},k}$, $\boldsymbol{T}_{\mathrm{ht},k}$, and $\boldsymbol{C}_{\mathrm{he},k}$ can be denoted by $\boldsymbol{L}_{\mathrm{sp},k} = \{(X_{k,1}, Y_{k,1}, H_{k,1}), (X_{k,2}, Y_{k,2}, H_{k,2}), \ldots, (X_{n_{\mathrm{sd},k}}, Y_{n_{\mathrm{sd},k}}, H_{n_{\mathrm{sd},k}})\}$, $\boldsymbol{T}_{\mathrm{ht},k} = \{t_{\mathrm{ht},k,1}, t_{\mathrm{ht},k,2}, \ldots, t_{\mathrm{ht},k,n_{\mathrm{sd}}}\}$, and $\boldsymbol{C}_{\mathrm{ht},k} = \{C_{\mathrm{ht},k,1}, C_{\mathrm{ht},k,2}, \ldots, C_{\mathrm{ht},k,n_{\mathrm{sd}}}\}$, respectively. The computing method of $t_{\mathrm{ctt},k}$ is shown in Algorithm 2, which is described as follows. Firstly, the distances between stop points and the DCS of UAV k are obtained. Then, all stop points are sorted in order from closest to farthest according to the distances. Lastly, UAV k collects the data sequentially according to the order of the sorted stop points. Once the energy of UAV k cannot complete the collection task at the next stop point, it will return to its DCS for charging.

D. Objective function

From the above description, the characteristics of the three stages shown in Fig. 1 can be summarized as follows:

- In stage 1, a UAV flies from the DC to its DCS, whose flight is constrained by its battery capability and a limited number of charging piles (it needs to compete with other UAVs for charging resources) at one node.
- In stage 2, a UAV needs to be recharged multiple times to complete the data collection task at multiple stop points. Therefore, once DCSs are identified, the impact of stage 1 on stage 2 includes: 1) stage 1 determines the start time of stage 2 and 2) when multiple UAVs share the same DCS and their flights determined in stage 1 consume similar

travel time, they may need to compete with each other for charging resources in stage 2.
- Stage 3 has the same characteristics as stage 1. Note that, the travel time of a UAV in stage 1 and stage 2 directly affects its travel time in stage 3.

Given these characteristics, the considered deployment and FP problem can be seen as a three-layer optimization problem:

- The first layer is to determine the optimal FP of UAVs in stage 1, which is measured by the average travel time of all UAVs. Let $\boldsymbol{L}_{\mathrm{a}}^{*}$ denote the optimal $\boldsymbol{L}_{\mathrm{a}}$. Let t_{adc} denote the average travel time of all UAVs in stage 1. To find $\boldsymbol{L}_{\mathrm{a}}^{*}$, the objective function can be defined by:

$$\text{Minimize } t_{\mathrm{adc}} = f(\boldsymbol{L}_{\mathrm{a},k}) = \frac{1}{n_{\mathrm{u}}} \sum_{k=1}^{n_{\mathrm{u}}} t_{\mathrm{fdc},k}, \tag{25}$$

subject to

$$\boldsymbol{L}_{\mathrm{a},k} \in \boldsymbol{M}_{\mathrm{CSN}}, k \in [1, n_{\mathrm{u}}]. \tag{26}$$

- The second layer determines the optimal deployment of UAVs in stage 2, which aims to search for the optimal average travel time of all UAVs. Let t_{act} denote the consumed average time, which can be defined by:

$$\begin{aligned}\text{Minimize } t_{\mathrm{act}} &= f(D_{\mathrm{cs},k}, \boldsymbol{L}_{\mathrm{sp},k}, \boldsymbol{T}_{\mathrm{ht},k}, \boldsymbol{C}_{\mathrm{he},k}) \\ &= \frac{1}{n_{\mathrm{u}}} \sum_{k=1}^{n_{\mathrm{u}}} t_{\mathrm{ctt},k},\end{aligned} \tag{27}$$

subject to (17), (18), (19), (20), and

$$X_{\mathrm{min},k,j} \le X_{k,j} \le X_{\mathrm{max},k,j}, j \in [1, n_{\mathrm{sp},k}], \tag{28}$$

$$Y_{\mathrm{min},k,j} \le Y_{k,j} \le Y_{\mathrm{max},k,j}, j \in [1, n_{\mathrm{sp},k}], \tag{29}$$

$$Z_{\mathrm{min},k,j} \le Z_{k,j} \le Z_{\mathrm{max},k,j}, j \in [1, n_{\mathrm{sp},k}], \tag{30}$$

where $X_{\mathrm{min},k,j}$, $Y_{\mathrm{min},k,j}$, and $Z_{\mathrm{min},k,j}$ are the lower limits of the coordinate values of UAV k at stop point j in the a-axis, y-axis, and z-axis, respectively; $X_{\mathrm{max},k,j}$, $Y_{\mathrm{max},k,j}$, and $Z_{\mathrm{max},k,j}$ are the upper limits of the coordinate values of UAV k at stop point j in the x-axis, y-axis, and z-axis, respectively.
- The third layer is to determine the optimal FP of all UAVs in stage 3, which is measured by the average travel time of all UAVs. Let $\boldsymbol{L}_{\mathrm{c}}^{*}$ denote the optimal $\boldsymbol{L}_{\mathrm{c}}$. Let t_{acd} denote the average travel time of all UAVs in stage 3. To find $\boldsymbol{L}_{\mathrm{c}}^{*}$, the objective function can be defined by:

$$\text{Minimize } t_{\mathrm{acd}} = f(\boldsymbol{L}_{\mathrm{c},k}) = \frac{1}{n_{\mathrm{u}}} \sum_{k=1}^{n_{\mathrm{u}}} t_{\mathrm{fcd},k}, \tag{31}$$

subject to

$$\boldsymbol{L}_{\mathrm{c},k} \in \boldsymbol{M}_{\mathrm{CSN}}, k \in [1, n_{\mathrm{u}}]. \tag{32}$$

Let T_{att} denote the average travel time of all UAVs in the three stages. The considered problem can be formulated as:

$$\text{Minimize } T_{\mathrm{att}} = t_{\mathrm{adc}} + t_{\mathrm{act}} + t_{\mathrm{acd}}, \tag{33}$$

subject to (17), (18), (19), (20), (26), (28), (29), (30), and (32).

Algorithm 2 Calculate $t_{\mathrm{ctt},k}$

Input:
$D_{\mathrm{cs},k}$, $\boldsymbol{L}_{\mathrm{sp},k}$, $\boldsymbol{T}_{\mathrm{ht},k}$, and $\boldsymbol{C}_{\mathrm{he},k}$
Output:
$t_{\mathrm{ctt},k}$
Initialize $t_{\mathrm{ctt},k}$ by $t_{\mathrm{ctt},k} = 0$;
Get $\boldsymbol{D}_{\mathrm{spc},k}$ (Euclidean distance between $D_{\mathrm{cs},k}$ and $\boldsymbol{L}_{\mathrm{sp},k}$);
Sort $\boldsymbol{D}_{\mathrm{spc},k}$ from small to large and get the index I_{d};
for $i = 1$ to n_{sd} **do**
  if $i == 1$ **then**
    $C_{\mathrm{sp},k,i} = \boldsymbol{C}_{\mathrm{he},I_{\mathrm{d},k,i}} + \frac{2\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}} P_0}{v}$;
  else
    Get D_{I} (Euclidean distance of $\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}}$ and $\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i-1}}$);
    $C_{\mathrm{sp},k,i} = \frac{D_{\mathrm{I}} P_0}{v} + \boldsymbol{C}_{\mathrm{he},k,I_{\mathrm{d},i}} + \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}} P_0}{v}$;
  end if
end for
$C_{\mathrm{R}} = C_0$;
for $i = 1$ to n_{sd} **do**
  $C_{\mathrm{TR}} = C_{\mathrm{R}} - C_{\mathrm{sp},k,i}$;
  if $C_{\mathrm{TR}} < 0$ **then**
    $C_{\mathrm{R}} = C_{\mathrm{R}} - \frac{(\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i-1}} + \boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}}) P_0}{v} - C_{\mathrm{tl}} + C_0 - \boldsymbol{C}_{\mathrm{he},k,I_{\mathrm{d},i}}$;
    Get charging time t_{tr} from $C_{\mathrm{R}} - \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i-1}} P_0}{v}$ to C_0 by (13);
    $t_{\mathrm{ctt},k} = t_{\mathrm{ctt},k} + \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i-1}}}{v} + \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}}}{v} + \boldsymbol{T}_{\mathrm{ht},k,I_{\mathrm{d},i}} + t_{\mathrm{tl}} + t_{\mathrm{tr}}$;
  else
    if $i == 1$ **then**
      $C_{\mathrm{R}} = C_{\mathrm{R}} - \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}} P_0}{v} - \boldsymbol{C}_{\mathrm{he},k,I_{\mathrm{d},i}}$;
      $t_{\mathrm{ctt},k} = t_{\mathrm{ctt},k} + \frac{\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}}}{v} + \boldsymbol{T}_{\mathrm{ht},k,I_{\mathrm{d},i}}$;
    else
      Get D_{I} (Euclidean distance of $\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i}}$ and $\boldsymbol{D}_{\mathrm{spc},k,I_{\mathrm{d},i-1}}$);
      $C_{\mathrm{R}} = C_{\mathrm{R}} - \frac{D_{\mathrm{I}} P_0}{v} - \boldsymbol{C}_{\mathrm{he},k,I_{\mathrm{d},i}}$;
      $t_{\mathrm{ctt},k} = t_{\mathrm{ctt},k} + \frac{D_{\mathrm{I}}}{v} + \boldsymbol{T}_{\mathrm{ht},k,I_{\mathrm{d},i}}$.
    end if
  end if
end for

Algorithm 3 The method of constructing path by the PBEM

Input:
$\boldsymbol{S}$ (the priority sequence), $\boldsymbol{V}_{\mathrm{p}}$ (the set of nodes corresponding to the constructed path), m_{N} (the number of nodes in the network), m_{s} (the source node), m_{d} (the destination node)
Output:
$\boldsymbol{V}_{\mathrm{p}}$
Initialize $\boldsymbol{V}_{\mathrm{p}}$ by $\boldsymbol{V}_{\mathrm{p}} = [m_{\mathrm{s}}]$ and update $\boldsymbol{S}$ by $\boldsymbol{S}(\boldsymbol{m}_{\mathrm{s}}) = -\infty$;
for $i = 1$ to m_{N} **do**
  Select node m_{k} that has the highest priority among the nodes having direct links with node $\boldsymbol{V}_{\mathrm{p}}(i)$;
  Update $\boldsymbol{V}_{\mathrm{p}}$ by $\boldsymbol{V}_{\mathrm{p}} = [\boldsymbol{V}_{\mathrm{p}}\ m_{\mathrm{k}}]$ and $\boldsymbol{S}$ by $\boldsymbol{S}(\boldsymbol{m}_{\mathrm{k}}) = -\infty$;
  if $m_{\mathrm{k}} == m_{\mathrm{d}}$ **then**
    Break.
  end if
end for

V. Methodology

To solve (33), EDDE-DPDE is proposed. This section first introduces DE and then EDDE-DPDE is presented.

A. DE

DE is a popular algorithm, which consists of three parts:

1) Mutation: The mutation operator aims at generating new individuals, which can be expressed by:

$$\boldsymbol{v}_i = \boldsymbol{x}_a + M_{\mathrm{f}}(\boldsymbol{x}_b - \boldsymbol{x}_c), i = 1, 2, \ldots, n_{\mathrm{p}}, \tag{34}$$

where n_{p} is the population size, $\boldsymbol{v}_i$ is the mutation vector of individual i, M_{f} is the mutation factor, and $\boldsymbol{x}_a$, $\boldsymbol{x}_b$, $\boldsymbol{x}_c$

Algorithm 4 The implementation of EDDE

Input:
$n_{\rm p}$ (population size), $n_{\rm u}$ (the number of UAVs), $n_{\rm n}$ (the number of nodes), $\boldsymbol{F}_{\rm fma}$ (the flight schedule in stage 1), $\boldsymbol{M}_{\rm CSN}$ (the connected matrix), and $N_{\rm Iter1}$ (the maximum number of iterations in stage 1)
Output:
$\boldsymbol{L}_{\rm a}^*$ and $t_{\rm adc}^*$
Initialize the population $\boldsymbol{X}_{\rm DC}$ by (41);
Compute $\boldsymbol{L}_{\rm a}$ corresponding to each individual in the $\boldsymbol{X}_{\rm DC}$ by PBEM;
Evaluate each $\boldsymbol{L}_{\rm a}$ by (1-16), and (25), $\boldsymbol{M}_{\rm CSN}$, and $\boldsymbol{F}_{\rm fma}$;
Find $\boldsymbol{x}^*{}_{\rm dc}$, $\boldsymbol{L}_{\rm a}^*$, and $t_{\rm adc}^*$;
Initialize $\boldsymbol{X}_{\rm E}$ by $\boldsymbol{X}_{\rm E} = \boldsymbol{x}^*{}_{\rm dc}$ and $L_{\rm E}$ (the length of $\boldsymbol{X}_{\rm E}$) by $L_{\rm E} = 1$;
for $l = 1$ to $N_{\rm Iter1}$ **do**
  Compute $M_{\rm p}$ by (38);
  for $j = 1$ to $n_{\rm p}$ **do**
    Initialize the flag bit $f_{\rm p}$ by $f_{\rm p} = 0$;
    Generate a, b, and c randomly;
    Perform the mutation operator by (37);
    Perform the crossover operator by (39);
    Compute $\boldsymbol{L}_{\rm a}$ corresponding to $\boldsymbol{v}_i$ by PBEM;
    Evaluate $\boldsymbol{L}_{\rm a}$ by (1-16), and (25), $\boldsymbol{M}_{\rm CSN}$, and $\boldsymbol{F}_{\rm fma}$;
    Perform the selection crossover by (36) and update $f_{\rm p}$ (if $\boldsymbol{v}_i$ is better than $\boldsymbol{X}_{{\rm dc},i}$, $f_{\rm p} = 1$);
    Update $\boldsymbol{x}^*{}_{\rm dc}$, $\boldsymbol{L}_{\rm a}^*$, and $t_{\rm adc}^*$;
    if $f_{\rm p} == 1$ **then**
      if $L_{\rm E} \le n_{\rm p}$ **then**
        Add $\boldsymbol{v}_i$ to $\boldsymbol{X}_{\rm E}$ and update $L_{\rm E}$ by $L_{\rm E} = L_{\rm E} + 1$;
      else
        Use $\boldsymbol{v}_i$ to replace randomly an individual of $\boldsymbol{X}_{\rm E}$;
      end if
    end if
    Perform local escape operator by (1-16), (25), and (40);
    Compute $\boldsymbol{L}_{\rm a}$ corresponding to $\boldsymbol{x}_{\rm p}^*$ by PBEM;
    Evaluate $\boldsymbol{L}_{\rm a}$ by (1-16), and (25), $\boldsymbol{M}_{\rm CSN}$, and $\boldsymbol{F}_{\rm fma}$;
    Update $\boldsymbol{x}^*{}_{\rm dc}$, $\boldsymbol{L}_{\rm a}^*$, and $t_{\rm adc}^*$.
  end for
end for

are three selected randomly individuals from the population $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_{\rm p}}\}$, where a, b, and c are three selected integers randomly between 1 and $n_{\rm p}$.

2) Crossover: The crossover operator is to generate the trial vector $\boldsymbol{z}_i$ of $\boldsymbol{v}_i$, which is represented by:

$$\boldsymbol{z}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, \text{ if } d \le C_{\rm f} || j = e; \\ \boldsymbol{x}_{i,j}, \text{ otherwise}, \end{cases} \tag{35}$$

where $\boldsymbol{z}_{i,j}$ is the jth variable of $\boldsymbol{z}_i$, $\boldsymbol{v}_{i,j}$ is the jth variable of $\boldsymbol{v}_i$, $\boldsymbol{x}_{i,j}$ is the jth variable of individual i, d is a random number between 0 and 1, $C_{\rm f}$ is the crossover factor, and e is a random integer between 1 and $n_{\rm p}$. In addition, j meets $j \in [1, D]$, where D is the number of variables.

3) Selection operator: The selection operator is to find a better solution between $\boldsymbol{z}_i$ and $\boldsymbol{x}_i$, which can be denoted by:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{z}_i, \text{ if } f(\boldsymbol{z}_i) \le f(\boldsymbol{x}_i); \\ \boldsymbol{x}_i, \text{ otherwise}. \end{cases} \tag{36}$$

B. The proposed EDDE-DPDE

EDDE-DPDE is based on the proposed EDDE and DPDE. Next, the proposed EDDE and DPDE are first introduced. Then, EDDE-DPDE is presented.

1) EDDE: EDDE is proposed to search $\boldsymbol{L}_{\rm a}^*$ and $\boldsymbol{L}_{\rm c}^*$ from $\boldsymbol{M}_{\rm CSN}$. To achieve this, this paper adopts a priority-based encoding mechanism (PBEM) [29] as shown in Algorithm 3, whose basic idea is to use a PS to generate a $\boldsymbol{L}_{{\rm a},k}$ or $\boldsymbol{L}_{{\rm c},k}$. In terms of population, each individual consists of $n_{\rm u}$ modules and each module denotes a PS (the length of a PS is equal to the number of nodes in the CSN) of a UAV, which can be expressed as follows: $\boldsymbol{X}_{\rm DC} = \{\boldsymbol{X}_{{\rm dc},1}, \boldsymbol{X}_{{\rm dc},2}, \ldots, \boldsymbol{X}_{{\rm dc},n_{\rm p}}\}$ and $\boldsymbol{X}_{{\rm dc},i} = \{\boldsymbol{x}_{{\rm dc},i,1}, \boldsymbol{x}_{{\rm dc},i,2}, \ldots, \boldsymbol{x}_{{\rm dc},i,n_{\rm u}}\}$, where $\boldsymbol{X}_{\rm DC}$ is a population with $n_{\rm p}$ individuals, $\boldsymbol{x}_{{\rm dc},i,k} (k \in [1, n_{\rm u}])$ is the kth module of individual i in the $\boldsymbol{X}_{\rm dc}$, and $x_{{\rm dc},i,k,j} (k \in [1, n_{\rm u}]$ and $j \in [1, n_{\rm n}])$ is the jth element of the kth module of individual i. Next, taking $\boldsymbol{X}_{\rm DC}$ as an example to introduce EDDE. EDDE consists of three core components:

- Mutation operator driven by the elite archive. To improve the directionality of the mutation, an elite archive $\boldsymbol{X}_{\rm E}$ is created to save some promising individuals, whose length is equal to $n_{\rm e}$. That is, $\boldsymbol{X}_{\rm E} = \{\boldsymbol{X}_{{\rm e},1}, \boldsymbol{X}_{{\rm e},2}, \ldots, \boldsymbol{X}_{{\rm e},n_{\rm p}}\}$. The designed mutation operator can be expressed by:

$$\boldsymbol{v}_i = \boldsymbol{X}_{{\rm e},a} + \varphi_1(\boldsymbol{X}_{{\rm dc},b} - \boldsymbol{X}_{{\rm dc},c}) + \varphi_2(\boldsymbol{X}_{{\rm e},a} - M_{\rm p}), \tag{37}$$

  where φ_1 and φ_2 are two random numbers between 0 and 1 (φ_1 and φ_2 meet $\varphi_1 + \varphi_2 = 1$), and $M_{\rm p}$ is the mean position of $\boldsymbol{X}_{\rm DC}$. $M_{\rm p}$ is computed by:

$$M_{\rm p} = \frac{1}{n_{\rm p}} \sum_{k=1}^{n_{\rm p}} X_{{\rm dc},i}. \tag{38}$$

  From (37) and (38), the created elite archive can guide the direction of mutation. In addition, the designed mutation operator also can be seen as a random mutation operator that uses random numbers to balance the local mutation factor (the second term on the right of (37)) and the global mutation factor (the third term on the right of (37)), which can help to keep the population diversity.
- Enhanced crossover operator. An individual with a better fitness value usually contains more valuable information to search for the global optimal solution. To make full use of the obtained best individual $\boldsymbol{x}^*{}_{\rm dc}$, an enhanced crossover operator is introduced, which is expressed by:

$$\boldsymbol{z}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, \text{ if } d \le 1/3; \\ \boldsymbol{x}^*{}_{{\rm dc},j}, \text{ if } d \ge 2/3; \quad i \in [1, n_{\rm p}], j \in [1, n_{\rm u} n_{\rm n}]. \\ \boldsymbol{x}_{{\rm dc},i,j}, \text{ otherwise}, \end{cases} \tag{39}$$

  From (39), $\boldsymbol{v}_{i,j}$, $\boldsymbol{x}^*{}_{{\rm dc},j}$, and $\boldsymbol{x}_{{\rm dc},i,j}$ have the same importance to the designed crossover operator, which does not refer to $C_{\rm f}$ that can help to the stability of EDDE.
- Local escape operator. To increase the chance of EDDE to escape from the local optima, a local escape operator is designed, whose basic idea is to use random numbers generated by opposition-based learning [30] to replace some elements of some modules of $\boldsymbol{x}^*{}_{\rm dc}$. The local escape operator can be expressed by:

$$x^*{}_{{\rm dc},j,o} = -x^*{}_{{\rm dc},j,o}, j \in [1, \lceil \theta n_{\rm u} \rceil], o \in [1, \lceil \eta n_{\rm n} \rceil], \tag{40}$$

  where θ and η are two random numbers between 0 and 1, j and o are two integers.

Like [29], $x_{{\rm dc},i,k,j}$ meets $x_{{\rm dc},i,k,j} \in [-n_{\rm n}, n_{\rm n}]$ in using PBEM to generate $\boldsymbol{L}_{{\rm a},k}$. Specifically, $\boldsymbol{X}_{\rm DC}$ is initialized by:

$$x_{{\rm dc},i,k,j} = 2 n_{\rm n} \mu - n_{\rm n}, \tag{41}$$

where $i \in [1, n_{\rm p}]$, $k \in [1, n_{\rm u}]$, $j \in [1, n_{\rm n}]$, and μ is a random number between 0 and 1. Algorithm 4 presents the computing method to obtain $\boldsymbol{L}_{\rm a}^*$ and $t_{\rm adc}^*$ (the optimal $t_{\rm adc}$) by EDDE.

Algorithm 5 Dynamic population mechanism

Input:
$\boldsymbol{X}_{\mathrm{D},k}$ (the population of UAV k), $n_{\mathrm{sp},k}$ (the number of stop points of UAV k), $\boldsymbol{z}$ (trail vector)
Output:
$\boldsymbol{X}^*_{\mathrm{D},k}$ (the optimal $\boldsymbol{X}_{\mathrm{D},\boldsymbol{k}}$)
1: **for** $l = i$ to n_{sp} **do**
2: Get $\boldsymbol{X}_{\mathrm{ID}}$ by inserting randomly $\boldsymbol{z}_i$ to $\boldsymbol{X}_{\mathrm{D},k}$;
3: Get $\boldsymbol{X}_{\mathrm{RD}}$ by using $\boldsymbol{z}_i$ to replace randomly an element of $\boldsymbol{X}_{\mathrm{D},k}$;
4: Get $\boldsymbol{X}_{\mathrm{MD}}$ by removing randomly an element of $\boldsymbol{X}_{\mathrm{D},k}$;
5: Evaluate $\boldsymbol{X}_{\mathrm{ID}}$, $\boldsymbol{X}_{\mathrm{RD}}$, and $\boldsymbol{X}_{\mathrm{MD}}$ by (17-24) and Algorithm 2;
6: Update $\boldsymbol{X}_{\mathrm{D},k}$ by choosing the best of $\boldsymbol{X}_{\mathrm{ID}}$, $\boldsymbol{X}_{\mathrm{RD}}$, and $\boldsymbol{X}_{\mathrm{MD}}$;
7: **end for**
8: Get $\boldsymbol{X}^*_{\mathrm{D},k}$ by $\boldsymbol{X}^*_{\mathrm{D},k}$=$\boldsymbol{X}_{\mathrm{D},k}$.

2) DPDE: DPDE is to compute the optimal number and locations of stop points of UAV k to complete the assigned data collection task. As done in [10], $\boldsymbol{X}_{\mathrm{D},k}$ in DPDE denotes an entire deployment of UAV k. $\boldsymbol{X}_{\mathrm{D},k}$ can be expressed by $\boldsymbol{X}_{\mathrm{D},k} = \{\boldsymbol{X}_{\mathrm{d},k,1}, \boldsymbol{X}_{\mathrm{d},k,2}, \ldots, \boldsymbol{X}_{\mathrm{d},k,n_{\mathrm{sp},k}}\}$, where $\boldsymbol{X}_{\mathrm{d},k,i} = \{(X_{k,i}, Y_{k,i}, H_{k,i})\}, i \in [1, n_{\mathrm{sp},k}]$. DPDE includes three parts:

- Mutation operator based on the historical population. As shown in (34), there is only a mutation strategy, which does not help to keep the population diversity [31]. Given this, a set of mutation operators with historical population is designed, which can be written by:

$$\boldsymbol{v}_i = \begin{cases} \boldsymbol{X}_{\mathrm{d},k,a} + \phi_1(\boldsymbol{X}_{\mathrm{d},k,b} - \boldsymbol{X}_{\mathrm{d},k,c}) + \\ \phi_2(\boldsymbol{X}_{\mathrm{d},k,i} - \boldsymbol{M}_{\mathrm{d}}), \text{if } \kappa \le \frac{1}{3}, \\ \boldsymbol{X}_{\mathrm{d},k,a} + \phi_1(\boldsymbol{X}_{\mathrm{h},k,g} - \boldsymbol{X}_{\mathrm{d},k,c}) + \\ \phi_2(\boldsymbol{X}_{\mathrm{d},k,i} - \boldsymbol{M}_{\mathrm{d}}), \text{if } \kappa \le \frac{2}{3}, \\ \boldsymbol{X}_{\mathrm{d},k,a} + \phi_1(\boldsymbol{X}_{\mathrm{h},k,g} - \boldsymbol{X}_{\mathrm{h},k,r}) + \\ \phi_2(\boldsymbol{X}_{\mathrm{d},k,i} - \boldsymbol{M}_{\mathrm{d}}), \text{otherwise}, \end{cases} \tag{42}$$

where a, b, and c are three random numbers that meet a, b, and $c \in [1, n_{\mathrm{sp},k}]$, g and r are two random integers, ϕ_1 and ϕ_2 are two random numbers between 0 and 1 (ϕ_1 and ϕ_2 meet $\phi_1 + \phi_2 = 1$), $\boldsymbol{M}_{\mathrm{d}}$ is the mean position of $\boldsymbol{X}_{\mathrm{D},k}$, $\boldsymbol{X}_{\mathrm{h},k,g}$ and $\boldsymbol{X}_{\mathrm{h},k,r}$ are the gth and the rth elements of $\boldsymbol{X}_{\mathrm{H},k}$, respectively. Let $n_{\mathrm{h},k}$ denote the length of $\boldsymbol{X}_{\mathrm{H},k}$. g and r meet $g, r \in [1, n_{\mathrm{h},k}]$. $\boldsymbol{M}_{\mathrm{d}}$ can be computed by:

$$M_{\mathrm{d}} = \frac{1}{n_{\mathrm{sp},k}} \sum_{k=1}^{n_{\mathrm{sp},k}} X_{\mathrm{d},k,i}. \tag{43}$$

In (42), the three learning strategies have the same importance and are assigned the same selected probability. Like (37), (42) also can be seen as a random mutation operator, which employs random numbers to balance the local mutation factor (the second term on the right of each mutation strategy) and the global mutation factor (the third term on the right of each mutation strategy). This can help to increase the population diversity.
- Random crossover operator. Using random numbers to replace crossover factor is the basic idea of the designed random crossover operator that can be expressed by:

$$\boldsymbol{z}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, \text{ if } d_1 \le d_2 || j = e; \\ \boldsymbol{X}_{\mathrm{d},k,i,j}, \text{ otherwise.} \end{cases} \tag{44}$$

- Dynamic population mechanism. To obtain the optimal number and locations of UAV k, a dynamic population mechanism [10] is employed, which consists of one insert operator, one remove operator, and one replace operator. Algorithm 5 shows the flowchart of this mechanism.

Algorithm 6 The implementation of DPDE

Input:
n_{u} (the number of UAVs), M_{Iter2} (the maximum number of iterations in stage 2)
Output:
$\boldsymbol{t}^*_{\mathrm{act}}$ (the optimal $\boldsymbol{t}_{\mathrm{act}}$)
1: **for** $k = 1$ to n_{u} **do**
2: Initialize $\boldsymbol{X}_{\mathrm{D},k}$ and $\boldsymbol{X}_{\mathrm{H},k}$ by (28-30);
3: Evaluate $\boldsymbol{X}_{\mathrm{D},k}$ by (17-24) and Algorithm 2;
4: Update $\boldsymbol{X}^*_{\mathrm{D},k}$ by $\boldsymbol{X}^*_{\mathrm{D},k}$=$\boldsymbol{X}_{\mathrm{D},k}$;
5: **for** $i = 1$ to M_{Iter2} **do**
6: Generate a random number ω between 0 and 1;
7: **if** $\omega \le 0.5$ **then**
8: $\boldsymbol{X}_{\mathrm{H},k} = \boldsymbol{X}_{\mathrm{D},k}$;
9: **end if**
10: Update $\boldsymbol{X}_{\mathrm{H},k}$ by randomly sorting the elements in the $\boldsymbol{X}_{\mathrm{H},k}$;
11: Perform the mutation operator by (42) and (43);
12: Perform the crossover operator by (44);
13: Update $\boldsymbol{X}^*_{\mathrm{D},k}$ by Algorithm 4;
14: **end for**
15: Get $\boldsymbol{L}^*_{\mathrm{sp},k}$ (the optimal $\boldsymbol{L}_{\mathrm{sp},k}$) by $\boldsymbol{X}^*_{\mathrm{D},k}$ and then compute $\boldsymbol{t}^*_{\mathrm{ctt},k}$ (the optimal $\boldsymbol{t}_{\mathrm{ctt},k}$) by (17-24) and Algorithm 2;
16: **end for**
17: Get $\boldsymbol{t}^*_{\mathrm{act}}$ by (27).

Algorithm 7 The implementation of EDDE-DPDE

Input:
n_{p} (population size), n_{u} (the number of UAVs), n_{n} (the number of nodes), $\boldsymbol{M}_{\mathrm{CSN}}$ (the connected matrix), M_{Iter1} (the maximum number of iterations in stage 1), M_{Iter2} (the maximum number of iterations in stage 2), $\boldsymbol{F}_{\mathrm{fma}}$ (the flight schedule in stage 1), $\boldsymbol{F}_{\mathrm{fmd}}$ (the flight schedule in stage 3), M_{Iter3} (the maximum number of iterations in stage 3)
Output:
The optimal average travel time t^*_{att}
1: Initialize t^*_{att} by $t^*_{\mathrm{att}} = +\infty$;
2: **for** $l = i$ to M_{Iter} **do**
3: Get t^*_{adc} by Algorithm 3;
4: Get t^*_{act} by Algorithm 5;
5: Get t^*_{acd} by Algorithm 3;
6: Compute t_{att} by (33);
7: **if** $t_{\mathrm{att}} \le t^*_{\mathrm{att}}$ **then**
8: $t^*_{\mathrm{att}} = t_{\mathrm{att}}$.
9: **end if**
10: **end for**

In [10], $n_{\mathrm{sp},k}$ is initialized by $n_{\mathrm{sp},k} = n_{\mathrm{IoT},k}$. To accelerate the convergence, $n_{\mathrm{sp},k}$ is initialized by $n_{\mathrm{sp},k} = 0.5 n_{\mathrm{IoT},k}$. The implementation of DPDE has been presented in Algorithm 6.

3) EDDE-DPDE: Algorithm 7 shows the implementation of EDDE-DPDE. According to Algorithm 7, EDDE-DPDE has a three-layer structure: 1) the first one (line 3 in Algorithm 7) is to determine the optimal FP in stage 1, 2) the second one (line 4 in Algorithm 7) is to determine the optimal deployment in stage 2, and 3) the third one (line 5 in Algorithm 7) is to determine the optimal FP in stage 3.

VI. Results and Comparisons

A. Experiment Preparation

Fig. 2 shows the considered scenario of using multi-UAVs for long-distance data collection from large-scale IoT devices according to the geographic information of Hong Kong. The considered scenario consists of a DC, 20 CSs, and a data collection zone (which is surrounded by a black box). The method of determining DCSs is as follows.

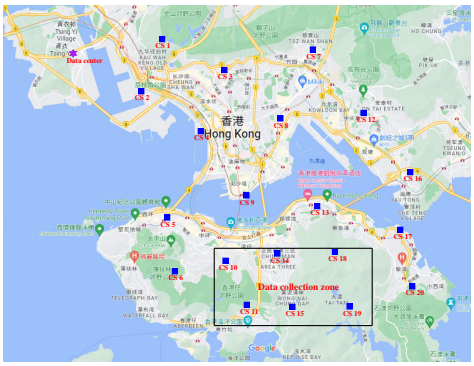


Fig. 2. The considered scenario of using multi-UAVs for long-distance data collection from large-scale IoT devices.

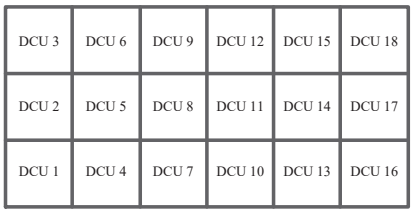


Fig. 3. The divided data collection units.

The data collection zone can be described by four pairs of longitude and latitude: (22.247781°N, 114.162139°E), (22.24778°N, 114.21976°E), (22.27475°N, 114.21976°E), and (22.27475°N, 114.16214°E). That is, the data collection zone can be regarded as a rectangle with a length of 6 Km and a width of 3 Km. Specifically, as shown in Fig. 3, the data collection zone can be divided into 18 data collection units (DCUs), and each DCU can be seen as a square with a side length of 1 Km. Each DCU corresponds to a UAV, whose DCS is the CS closest to the center of the DCU. Thus, 18 UAVs are required, whose DCSs are listed in Table II.

In addition, the parameters of a UAV and IoT devices are extracted from [10] and [25], which are as follows: $C_0 = 3.20 \times 10^5$J, $C_{\text{tl}} = 1.25 \times 10^4$J, $t_{\text{tl}} = 50$s, $v = 4$m/s, $P_0 = 300$W, $L_{\max} = 4.27$Km, the number of IoT devices in each DCU is set to 300, the amount of data of each IoT device is between 500 MB and 1500 MB, $G_0 = -30$dB, $\delta^2 = -250$dB, $n_{\text{sd}} = 10$, $B = 1$MHz, $P_{\text{hp}} = 1000$W, and $P_i = 0.1$W.

To verify the performance of EDDE-DPDE, it is compared with DE-DEVIPS, BA-DEVIPS, PSO-DEVIPS, and JAYA-DEVIPS as shown in Table III. In Table III, DE, bat algorithm (BA) [32], particle swarm optimization (PSO) [33], and Jaya algorithm (JAYA) [34] are four popular population-based metaheuristic algorithms. Like DPDE, differential evolution with a variable population size (DEPS) [10] is a also dynamic population differential evolution. In addition, the population size is set to 50 for EDDE, DE, BA, PSO, and JAYA. From Algorithm 4, each individual in EDDE is evaluated twice in one loop. To make a fair comparison, the maximum number of iterations is set to 250 and 500 for EDDE and the compared algorithms (i.e., DE, BA, PSO, and JAYA), respectively. The maximum number of iterations of DPDE and DEPS is set to 8,000. Other parameters of the compared algorithms are from the corresponding references. n_{c} is set to 3 at each node and

TABLE II: The task scheduler of UAVs.

No.	DCS	No.	DCS	No.	DCS
UAV 1	CS 11	UAV 7	CS 15	UAV 13	CS 19
UAV 2	CS 10	UAV 8	CS 14	UAV 14	CS 18
UAV 3	CS 10	UAV 9	CS 14	UAV 15	CS 18
UAV 4	CS 11	UAV 10	CS 15	UAV 16	CS 19
UAV 5	CS 11	UAV 11	CS 15	UAV 17	CS 19
UAV 6	CS 10	UAV 12	CS 14	UAV 18	CS 18

TABLE III: The structures of the applied algorithms.

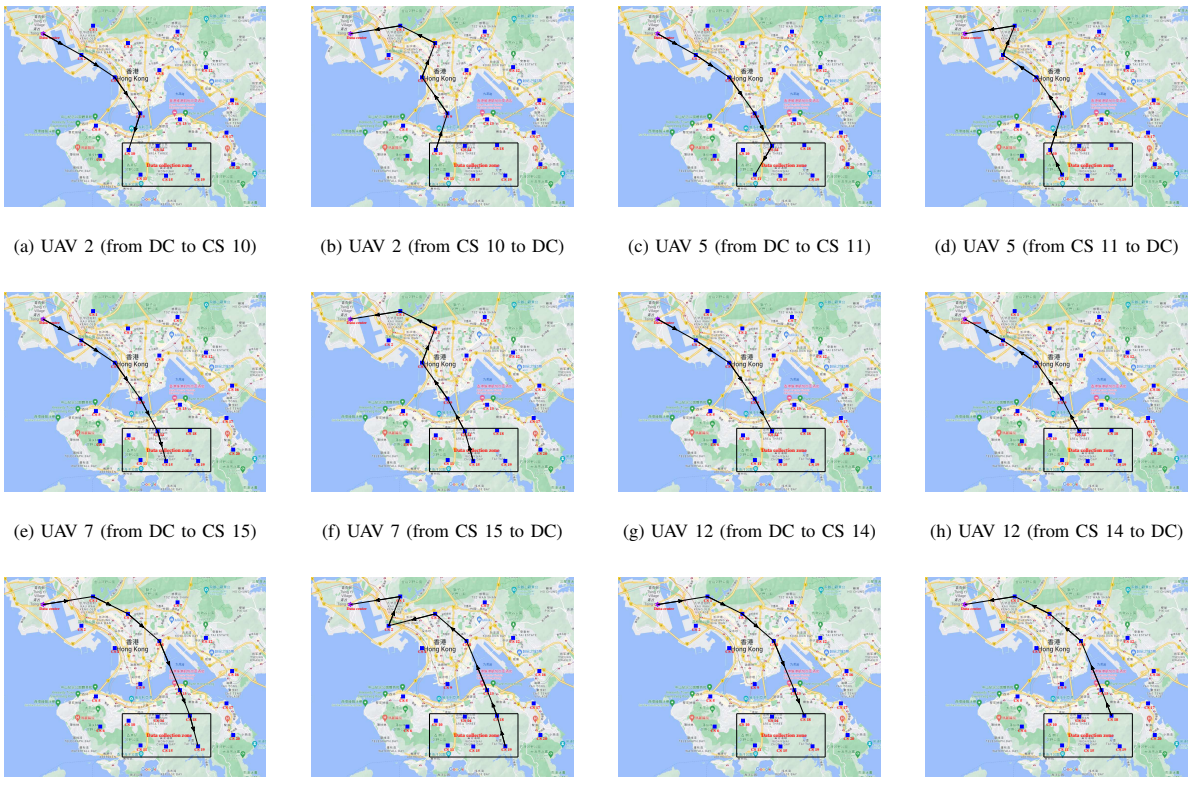
Algorithm	EDDE-DPDE	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS
Layer-1	EDDE	DE	BA	PSO	JAYA
Layer-2	DPDE	DEVIPS	DEVIPS	DEVIPS	DEVIPS
Layer-3	EDDE	DE	BA	PSO	JAYA

the number of independent runs for each algorithm is set to 30. Besides, the following quality indicators are employed:

- Value-based indicator. This indicator is to evaluate the solution accuracy, which includes the best solution (BEST), the mean solution (MEAN), the worst solution (WORST), the standard deviation (STD), and the average improvement rate (AIR). In terms of BEST, MEAN, and WORST, the smaller the value, the higher the accuracy of the solution obtained by an algorithm. A smaller STD means that an algorithm has stronger solution stability. Specifically, BEST, MEAN, WORST, and STD can be obtained by the statistical results of 30 independent runs. In addition, AIR_{AB} (the AIR of Algorithm A relative to Algorithm B) can be defined by:

$$\text{AIR}_{\text{AB}} = \frac{M_{\text{B}} - M_{\text{A}}}{M_{\text{A}}} \times 100\%, \tag{45}$$

  where M_{A} and M_{B} are the MEAN of Algorithm A and Algorithm B, respectively. From (45), the larger the value of AIR_{AB}, the greater the advantage of Algorithm A over Algorithm B. Tables IV-VII present the results of the value-based indicators, and the best results are in bold.
- Significance-based indicator. This indicator is to determine whether there are significant differences between the results obtained by the applied algorithms. To achieve this, a non-parametric Wilcoxon signed-rank test with a significance level $\alpha = 0.05$ [35] has been conducted for the mean results of 30 runs for each algorithm. The results produced by Wilcoxon signed-rank test have been displayed in Table VIII. In Table VIII, R^+ is the sum of positive rank, R^- is the sum of negative rank, P is the probability, and '+' means the null hypothesis is rejected ($P \leq 0.05$) with 95% certainty (the results of EDDE-DPDE and the compared algorithm have significant differences, and EDDE-DPDE shows better performance).
- Rank-based indicator. This indicator is to rank all algorithms. Friedman test [36] is a very powerful non-parametric statistical test, which is often used to compare the differences among algorithms [37], [38]. In this paper, the mean results of the applied algorithms are subjected to the Friedman test, and the Friedman mean rank (FMR) is recorded. Specifically, an algorithm with a smaller FMR means it has a better performance. The results of FMR have been shown in Table IX.



(a) UAV 2 (from DC to CS 10) (b) UAV 2 (from CS 10 to DC) (c) UAV 5 (from DC to CS 11) (d) UAV 5 (from CS 11 to DC)

(e) UAV 7 (from DC to CS 15) (f) UAV 7 (from CS 15 to DC) (g) UAV 12 (from DC to CS 14) (h) UAV 12 (from CS 14 to DC)

(i) UAV 13 (from DC to CS 19) (j) UAV 13 (from CS 19 to DC) (k) UAV 15 (from DC to CS 18) (l) UAV 15 (from CS 18 to DC)

Fig. 4. 12 optimal flights from six UAVs obtained by EDDE-DPDE (the direction of the arrow is the movement direction of the UAV between two nodes).

TABLE IV: The statistical results of average travel time from 30 independent runs (the unit is seconds).

Indicator	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS	EDDE-DPDE
BEST	40932.72	42652.12	42912.45	42587.22	**38083.76**
MEAN	44737.83	46577.89	46583.57	46115.18	**40265.08**
WORST	47568.01	50585.71	51197.28	50809.4	**42427.76**
STD	1641.803	2136.288	2126.387	2134.692	**1134.90**
AIR	11.11%	15.68%	15.69%	14.53%	–

TABLE V: The statistical results of average waiting time from 30 independent runs (the unit is seconds).

Indicator	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS	EDDE-DPDE
BEST	**1064.42**	1442.26	1147.91	1201.17	1083.79
MEAN	1808.08	2002.03	1721.78	1780.47	**1381.87**
WORST	2717.99	2763.24	2488.08	2654.08	**1775.76**
STD	395.86	376.41	370.45	357.14	**161.64**
AIR	30.84%	44.88%	24.59%	28.84%	–

TABLE VI: The statistical results of average charging time from 30 independent runs (the unit is seconds).

Indicator	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS	EDDE-DPDE
BEST	23139.60	23970.37	24122.31	23882.87	**21580.08**
MEAN	25411.36	26414.71	26488.58	26193.71	**23748.68**
WORST	27188.22	28897.30	29255.35	29312.66	**2441.39**
STD	1017.73	1263.38	1246.43	1308.29	**717.72**
AIR	7.00%	11.23%	11.54%	10.29%	–

TABLE VII: The statistical results of average straight flight time from 30 independent runs (the unit is seconds).

Indicator	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS	EDDE-DPDE
BEST	5484.75	5484.87	5485.67	5485.44	**5269.16**
MEAN	5848.37	5848.98	5849.14	5848.95	**5629.52**
WORST	6249.64	6250.62	6251.02	6251.01	**6033.56**
STD	855.74	856.31	856.25	856.52	**830.26**
AIR	3.89%	3.90%	3.90%	3.90%	–

B. Case 1: comparison on average travel time

Average travel time is the consumed average time by all UAVs to complete the task. Clearly, the smaller the consumed average travel time, the more efficient the algorithm.

The statistical results of average travel time obtained by five algorithms have been presented in Table IV. By observing Table IV, EDDE-DPDE is the best of all algorithms in terms of BEST, MEAN, WORST, and STD, which means that EDDE-DPDE shows stronger global search ability and stability. DE-DEPS also can provide a competitive solution, which achieves the second best BEST, MEAN, WORST, and STD. PSO-DEPS is the worst of all algorithms in terms of BEST, MEAN, and WORST. Specifically, in terms of AIR, EDDE-DPDE is better than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS by 11.11%, 15.68%, 15.69%, and 14.53%, respectively. This indicates that EDDE-DPDE can provide a more efficient solution to the considered joint optimization problem compared with DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS. In addition, as shown in the fourth row of

Table VIII, the test results of the Wilcoxon signed-rank test also support that EDDE-DPDE outperforms DE-DEPS, JAYA-DEPS, PSO-DEPS, and BA-DEPS. According to the obtained FMR in the second row of Table IX, all the applied algorithms can be sorted in the following order (from best to worst): EDDE-DPDE, DE-DEPS, JAYA-DEPS, PSO-DEPS, and BA-DEPS. EDDE-DPDE is the best of all algorithms. Besides, Fig. 4 presents 12 optimal flights from six UAVs that EDDE-DPDE searches from all candidate flights under the premise of consuming the least average travel time. By observing Fig. 4, the optimal flights of six UAVs obtained by EDDE-DPDE in stage 1 and stage 3 are not much different, which can be explained as follows:

- In stage 1, UAVs fly from the DC to six CSs at the same time and starting point. Thus, although UAVs fly towards six CSs, UAVs will face great competitive pressure for charging resources in the early stage of the flight.
- In stage 3, UAVs fly toward the DC. Note that, there is a certain time interval for the departure time of UAVs in stage 3 due to the spent different times by each UAV in stage 1 and stage 2, which can help to reduce the competitive pressure of UAVs for charging resources.
- The considered objective function focuses on the average travel time of all UAVs rather than a single UAV.

The above discussion supports the superiority of EDDE-DPDE in solving the considered joint optimization problem and the effectiveness of the improved strategies.

C. Case 2: comparison on average waiting time

Average waiting time is the consumed average time by all UAVs waiting for charging in the process of executing the task. Specifically, an efficient algorithm should reduce the consumed average waiting time as much as possible on the premise of completing the task.

Table V shows the statistical results of the average waiting time achieved by five algorithms. Looking at Table V, although DE-DEPS can get the optimal BEST, DE-DEPS cannot compete with EDDE-DPDE on MEAN, WORST, and STD. DE-DEPS and EDDE-DPDE can achieve similar solutions on BEST. Note that, DE-DEPS is the worst of all algorithms in terms of STD, which means the stability of DE-DEPS is the worst of all algorithms. In addition, EDDE-DPDE shows significant advantages over BA-DEPS, PSO-DEPS, and JAYA-DEPS on BEST, MEAN, WORST, and STD. Specifically, according to AIR, EDDE is better than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS by 30.84%, 44.88%, 24.59%, and 28.84%, respectively. That is, EDDE-DEPS is significantly superior to DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS in terms of saving the average waiting time. Besides, according to the fifth row of Table VIII, the test results from the Wilcoxon signed-rank test also prove that the compared algorithms are inferior to EDDE-DPDE. In terms of the obtained FMR presented in the third row of Table IX, all the applied algorithms can be sorted in the following order (from best to worst): EDDE-DPDE, PSO-DEPS, JAYA-DEPS, DE-DEPS, and BA-DEPS. EDDE-DPDE is the best of all algorithms.

From the above discussion, EDDE-DPDE shows obvious comprehensive advantages over DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS in terms of reducing average waiting time. This advantage is very helpful for EDDE-DPDE to be used for determining the flight planning of a UAV to execute a long-distance flight task under the UAV-station model.

D. Case 3: comparison on average charging time

Average charging time is the consumed average time by all UAVs charging in the process of executing the task. An efficient algorithm should reduce average charging time as much as possible on the premise of completing the task.

Table VI presents the statistical results of average charging time from five algorithms. From Table VI, EDDE-DPDE can get the optimal BEST, MEAN, WORST, and STD, which means EDDE-DPDE outperforms DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS in terms of global search ability and stability. DE-DEPS also shows excellent performance, which is second only to EDDE-DPDE in terms of BEST, MEAN, WORST, and STD. According to AIR, EDDE-DPDE is better than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS by 7.00%, 11.23%, 11.54%, and 10.29%, respectively. In addition, looking at the sixth row of Table VIII, the test results of the Wilcoxon signed-rank test determine that EDDE-DPDE can provide better solutions than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS. According to the obtained FMR shown in the fourth row of Table IX, all the applied algorithms can be sorted in the following order (from best to worst): EDDE-DPDE, DE-DEPS, JAYA-DEPS, PSO-DEPS, and BA-DEPS. EDDE-DPDE is the best of all algorithms.

When the UAV-station model is employed to add the energy of a UAV, charging time will take up a large proportion of the total travel time. As discussed above, EDDE-DPDE shows obvious advantages over the compared algorithms in terms of saving the charging time, which proves the great potential of EDDE-DPDE to be applied to determine the flight planning of a UAV to carry out a long-distance flight task under the UAV-station model.

E. Case 4: comparison on average straight flight time

Average straight flight time is the consumed average time by all UAVs on the straight flight during the task. An efficient algorithm should reduce average straight flight time as much as possible on the premise of completing the task.

The statistical results of average straight flight time from five algorithms have been shown in Table VII. As presented in Table VII, in terms of BEST, MEAN, WORST, and STD, EDDE-DPDE is the best of all algorithms, which proves its excellent global search ability and stability. DE-DEPS shows slight advantages over BA-DEPS, PSO-DEPS, and JAYA-DEPS on BEST, MEAN, WORST, and STD. From AIR, EDDE-DPDE is better than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS by 3.89%, 3.90%, 3.90%, and 3.90%, respectively. In addition, as displayed in the seventh row of Table VIII, the test results of the Wilcoxon signed-rank test support that DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS cannot compete with EDDE-DPDE. According to the

TABLE VIII: The results of Wilcoxon signed-rank test.

No.	EDDE-DPDE vs.											
	DE-DEPS			BA-DEPS			PSO-DEPS			JAYA-DEPS		
	R^+	R^-	P	R^+	R^-	P	R^+	R^-	P	R^+	R^-	P
Case 1	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$
Case 2	453	12	$5.75E-6^+$	465	0	$1.73E-6^+$	432	33	$4.07E-5^+$	450	15	$7.69E-6^+$
Case 3	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$
Case 4	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$	465	0	$1.73E-6^+$

TABLE IX: The obtained FMR by executing Friedman test.

No.	DE-DEPS	BA-DEPS	PSO-DEPS	JAYA-DEPS	EDDE-DPDE
Case 1	2.20	4.20	4.00	3.60	**1.00**
Case 2	3.37	4.13	2.87	3.17	**1.47**
Case 3	2.23	4.27	4.07	3.43	**1.00**
Case 4	2.33	3.90	4.10	3.67	**1.00**

obtained FMR listed in the fifth row of Table IX, all algorithms can be sorted in the following order (from best to worst): EDDE-DPDE, DE-DEPS, JAYA-DEPS, BA-DEPS, and PSO-DEPS. That is, EDDE-DPDE is the best of all algorithms.

Like charging time, the straight flight time also takes up a large proportion of the total travel time under the UAV-station model. From the above discussion, EDDE-DPDE shows better comprehensive performance than DE-DEPS, BA-DEPS, PSO-DEPS, and JAYA-DEPS in terms of reducing the average straight flight time. This supports the superiority of EDDE-DPDE in determining the flight planning of a UAV to execute a long-distance flight task under the UAV-station model.

VII. Conclusion

This paper aims to use multi-UAVs to collect data from large-scale IoT devices. We design a multi-UAVs-assisted large-scale IoT data collection system that can support long-distance data collection. To make the system work efficiently, a new population-based optimization algorithm, namely EDDE-DPDE, is reported to solve the joint optimization problem of FP and deployment for multi-UAVs. EDDE-DPDE has a three-layer structure. The first and third layers are EDDE, which is to determine the optimal FP of multi-UAVs between the DC and their DCSs. The second layer is DPDE, which is to optimize the deployment of multi-UAVs in the data collection zone. The EDDE is based on three improved strategies: mutation operator driven by the created elite archive, enhanced crossover operator, and local escape operator. The DPDE is based on two improved strategies: mutation operator with historical population and random crossover operator. To investigate the performance of EDDE-DPDE, a data collection scenario based on real geographic information is designed. Numerical results demonstrate that EDDE-DPDE shows an improvement of at least 11.11% compared with four powerful algorithms in terms of average travel time, which proves the validity of the improved strategies.

In this paper, data collection units are divided based on the area of the data collection zone. In future research, we will pay attention to designing an adaptive task allocation method based on the data amount and location distribution of IoT devices to further improve the work efficiency of this system.

References

[1] S. H. Alsamhi, F. A. Almalki, O. Ma, M. S. Ansari, and B. Lee, "Predictive estimation of optimal signal strength from drones over IoT frameworks in smart cities," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 402–416, 2023.

[2] G. Raja, S. Anbalagan, A. Ganapathisubramaniyan, M. S. Selvakumar, A. K. Bashir, and S. Mumtaz, "Efficient and secured swarm pattern multi-UAV communication," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7050–7058, 2021.

[3] T. Tang, T. Hong, H. Hong, S. Ji, S. Mumtaz, and M. Cheriet, "An improved UAV-PHD filter-based trajectory tracking algorithm for multi-UAVs in future 5G IoT scenarios," *Electronics*, vol. 8, no. 10, 2019.

[4] A. Andreou, C. X. Mavromoustakis, J. M. Batalla, E. K. Markakis, G. Mastorakis, and S. Mumtaz, "UAV trajectory optimisation in smart cities using modified A* algorithm combined with haversine and vincenty formulas," *IEEE Transactions on Vehicular Technology*, 2023.

[5] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 73–82, 2017.

[6] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6898–6908, 2020.

[7] J. Wang, C. Jiang, Z. Wei, C. Pan, H. Zhang, and Y. Ren, "Joint UAV hovering altitude and power control for space-air-ground IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1741–1753, 2019.

[8] X. Li, H. Yao, J. Wang, S. Wu, C. Jiang, and Y. Qian, "Rechargeable multi-UAV aided seamless coverage for QoS-guaranteed IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 902–10 914, 2019.

[9] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, "Energy-efficient data collection and device positioning in UAV-assisted IoT," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1122–1139, 2020.

[10] P.-Q. Huang, Y. Wang, K. Wang, and K. Yang, "Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 324–335, 2020.

[11] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 34–46, 2020.

[12] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "Aoi-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1211–1223, 2021.

[13] Y. Li, W. Liang, W. Xu, Z. Xu, X. Jia, Y. Xu, and H. Kan, "Data collection maximization in IoT-sensor networks via an energy-constrained UAV," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 159–174, 2023.

[14] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Offline and online UAV-enabled data collection in time-constrained IoT networks," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1918–1933, 2021.

[15] R. Liu, Z. Qu, G. Huang, M. Dong, T. Wang, S. Zhang, and A. Liu, "DRL-UTPS: DRL-based trajectory planning for unmanned aerial vehicles for data collection in dynamic IoT network," *IEEE Transactions on Intelligent Vehicles*, pp. 1–14, 2022.

[16] X. Wang, M. C. Gursoy, T. Erpek, and Y. E. Sagduyu, "Learning-based UAV path planning for data collection with integrated collision avoidance," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 663–16 676, 2022.

[17] Z. Na, M. Zhang, J. Wang, and Z. Gao, "UAV-assisted wireless powered Internet of Things: Joint trajectory optimization and resource allocation," *Ad Hoc Networks*, vol. 98, p. 102052, 2020.

[18] D.-H. Tran, V.-D. Nguyen, S. Chatzinotas, T. X. Vu, and B. Ottersten, "UAV relay-assisted emergency communications in IoT networks: Resource allocation and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1621–1637, 2022.

[19] Z. Na, Y. Liu, J. Shi, C. Liu, and Z. Gao, "UAV-supported clustered noma for 6G-enabled Internet of Things: Trajectory planning and resource allocation," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15 041–15 048, 2021.

[20] S. Han, K. Zhu, M. Zhou, and X. Liu, "Joint deployment optimization and flight trajectory planning for UAV assisted IoT data collection: A bilevel optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 492–21 504, 2022.

[21] L. Dong, Z. Liu, F. Jiang, and K. Wang, "Joint optimization of deployment and trajectory in UAV and IRS-assisted IoT data collection system," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 583–21 593, 2022.

[22] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2020.

[23] H. Pan, Y. Liu, G. Sun, J. Fan, S. Liang, and C. Yuen, "Joint power and 3D trajectory optimization for UAV-enabled wireless powered communication networks with obstacles," *IEEE Transactions on Communications*, pp. 1–1, 2023.

[24] P. W. Shaikh, M. El-Abd, M. Khanafer, and K. Gao, "A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 48–63, 2022.

[25] C. Huang, Z. Ming, and H. Huang, "Drone stations-aided beyond-battery-lifetime flight planning for parcel delivery," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2022.

[26] Y. Pan, Q. Chen, N. Zhang, Z. Li, T. Zhu, and Q. Han, "Extending delivery range and decelerating battery aging of logistics UAVs using public buses," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[27] J. Kim, Y. Choi, S. Jeon, J. Kang, and H. Cha, "Optrone: Maximizing performance and energy resources of drone batteries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3931–3943, 2020.

[28] B. Do Valle, C. T. Wentz, and R. Sarpeshkar, "An area and power-efficient analog Li-ion battery charger circuit," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 2, pp. 131–137, 2011.

[29] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing*, vol. 8, no. 4, pp. 1643–1653, 2008.

[30] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[31] H. Rakhshani and A. Rahati, "Snap-drift cuckoo search: A novel cuckoo search optimization algorithm," *Applied Soft Computing*, vol. 52, pp. 771–794, 2017.

[32] X.-S. Yang, "Chapter 10 - bat algorithms," in *Nature-Inspired Optimization Algorithms*, X.-S. Yang, Ed. Oxford: Elsevier, 2014, pp. 141–154.

[33] Z. Ren, A. Zhang, C. Wen, and Z. Feng, "A scatter learning particle swarm optimization algorithm for multimodal problems," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1127–1140, 2014.

[34] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[35] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[36] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

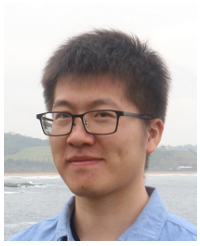
[37] H. Liu, J. Zhang, and M. Zhou, "Adaptive particle swarm optimizer combining hierarchical learning with variable population," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 3, pp. 1397–1407, 2023.

[38] Y. Pan, K. Gao, Z. Li, and N. Wu, "Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 361–371, 2023.

Yiying Zhang received the B.S. degree from Yanshan University, Qinhuangdao, China, in 2014, the M.S. degree from Harbin Institute of Technology, Harbin, China, in 2016, and the Ph.D. degree from Tianjin University, Tianjin, China, in 2021, all in information and communication engineering. He is currently a postdoctoral fellow at the Hong Kong Polytechnic University. His research interests include machine learning, evolutionary computation, and path planning of unmanned aerial vehicles.

Yue Huang is a Ph.D. candidate in the department of Electrical and Computer Engineering at the University of British Columbia. Her research interests span many areas of usable security and privacy, including mobile and IoT security, security and privacy in online social networks, and web security. Her work has appeared in a diverse set of top-tier venues on security and human–computer interaction, including ACM CHI, IEEE S&P, USENIX Security, CSCW, and SOUPS.

Chao Huang received the Ph.D. degree in control engineering from the University of Wollongong, Wollongong, NSW, Australia, in 2018. She is currently a Research Assistant Professor with the Department of Industrial and System Engineering, The Hong Kong Polytechnical University (PolyU), Hong Kong. Prior to PolyU, she was a Research Fellow with Nanyang Technological University (NTU), Singapore, and the National Institutes of Informatics (NII), Tokyo, Japan. Her research interests include human–machine collaboration, fault-tolerant control, mobile robot (EV, UAV), and path planning and control.



Hailong Huang received the Ph.D degree in Systems and Control from the University of New South Wales, Sydney, Australia, in 2018. He was a postdoctoral research fellow at the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia. He is now an Assistant Professor at the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong. His current research interests include guidance, navigation, and control of UAVs and mobile robots.

Anh-Tu Nguyen received the degree in engineering and the M.Sc. degree in automatic control from the Grenoble Institute of Technology, Grenoble, France, in 2009, and the Ph.D. degree in automatic control from the University of Valenciennes, Valenciennes, France, in 2013. He is currently an Associate Professor with INSA Hauts-de-France, Université Polytechnique Hautsde-France, Valenciennes. His research interests include robust control and estimation, cybernetics control systems and human–machine shared control with a strong emphasis on mechatronics applications.