Northumbria Research Link

Northumbria
University
NEWCASTLE

Differential Evolution with a Variable Population Size for Deployment Optimization in a UAV-Assisted IoT Data Collection System

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

***Abstract*—This paper studies an unmanned aerial vehicle (UAV)-assisted Internet of Things (IoT) data collection system, where a UAV is employed as a data collection platform for a group of ground IoT devices. Our objective is to minimize the energy consumption of this system by optimizing the UAV's deployment, including the number and locations of stop points of the UAV. When using evolutionary algorithms to solve this UAV's deployment problem, each individual usually represents an entire deployment. Since the number of stop points is unknown *a priori*, the length of each individual in the population should be varied during the optimization process. Under this condition, the UAV's deployment is a variable-length optimization problem and the traditional fixed-length mutation and crossover operators should be modified. In this paper, we propose a differential evolution algorithm with a variable population size, called DEVIPS, for optimizing the UAV's deployment. In DEVIPS, the location of each stop point is encoded into an individual, and thus the whole population represents an entire deployment. Over the course of evolution, differential evolution is employed to produce offspring. Afterward, we design a strategy to adjust the population size according to the performance improvement. By this strategy, the number of stop points can be increased, reduced, or kept unchanged adaptively. In DEVIPS, since each individual has a fixed length, the UAV's deployment becomes a fixed-length optimization problem and the traditional fixed-length mutation and crossover operators can be used directly. The performance of DEVIPS is compared with that of five algorithms on a set of instances. The experimental studies demonstrate its effectiveness.**

***Index Terms*—Deployment optimization; UAV; encoding; variable population size; differential evolution.**

I. Introduction

Internet of Things (IoT) aims at enabling a massive number of limited-power devices to be connected into a large-scale interconnected network [1], [2]. In recent years, IoT has been successfully applied to various fields such as intelligent transportation [3], healthcare [4], smart cities [5], and smart grids [6]. However, IoT devices are generally incapable of transmitting over a long distance due to their energy limitations. For example, in areas where terrestrial wireless networks are poorly covered, limited-power IoT devices may not be able to transmit their data to a remote base station. Therefore, it is a challenging task to efficiently collect data from IoT devices [7].

Making use of unmanned aerial vehicles (UAVs) as emerging data collection platforms has received wide attention [8]–[10]. Compared with terrestrial data collection platforms, UAV-assisted platforms can bring several advantages: 1) Due to their agility and mobility, UAVs can fly flexibly toward target devices; 2) UAVs have a high possibility of establishing line-of-sight links to target devices; and 3) UAVs can provide emergency services to target devices in unexpected or temporary events. Hence, the use of UAVs is expected to provide a promising way to collect data from IoT devices.

In order to efficiently employ a UAV, the UAV's deployment must be optimized. If deployed properly, a UAV should be able to provide a reliable and energy-efficient data collection solution for IoT devices [11]. In fact, the UAV's deployment optimization has been studied before. For example, He *et al.* [12] investigated the joint altitude and beamwidth optimization problem in the case of downlink multicasting, downlink broadcasting, or uplink multiple access. Alzenad *et al.* [13] optimized the UAV's deployment with the aim of using minimum transmission power to cover the maximum number of ground users. They then researched deployment optimization with different quality of service requirements [14]. Fan *et al.* [15] studied UAV deployment for a UAV-assisted relay system, where a UAV is utilized as a relay for the data transmission between source nodes and destination nodes. Mozaffari *et al.* [9] designed the optimal deployment of multiple UAVs for data collection from IoT devices. Du *et al.* [16] studied the UAV's deployment in a downlink communication converge scenario. It is worth noting that the number of stop points is preset in these papers, which means that these papers only optimize the locations of stop points. However, if the preset number is not optimal, it is very likely to result in a sub-optimal deployment.

Unlike existing works, in this paper, we optimize the number and locations of stop points simultaneously for the UAV's deployment. Due to the fact that the number of stop points is unknown *a priori*, it should be variable during the optimization

This work was supported in part by the Innovation-Driven Plan in Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grants 61673397, 61976225, 61572389, and 61620106011, and in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06. (*Corresponding author: Yong Wang and Kezhi Wang*).

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

K. Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle NE1 8ST, UK. (Email: kezhi.wang@northumbria.ac.uk)

K. Yang is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (Email: kunyang@essex.ac.uk)

process, which poses a great challenge to traditional gradient-based approaches as there is no explicit definition of a gradient vector. Thanks to the gradient-free nature of evolutionary algorithms (EAs), they have the potential to address this issue. EAs work with a population of individuals and each individual usually represents a candidate solution (i.e., an entire deployment). Because of the variable number of stop points in this paper, the length of each individual in the population is not fixed. As pointed out in [17], under this condition, it is necessary to modify the traditional fixed-length mutation and crossover operators in EAs, such as the n-point (most commonly, one-point or two-point) crossover operator[1] in genetic algorithm (GA) and the mutation operator "DE/rand/1" in differential evolution (DE). To this end, two kinds of attempts have been made in previous studies:

- *Designing special mutation and crossover operators:* One representative is the cut-and-splice operator. It is similar to the commonly used n-point crossover operator, but the locations of crossover points of two individuals do not have to be the same [17]. So far, this operator has been applied in wireless transmitter placement [18], long term evolution network planning [19], the carpool service problem [20], and pixel classification [21]. The spatial crossover operator is another representative, which works in the phenotypic space. In [22], a square area is divided into two halves along one of the dimensions, and then two halves from two individuals are stitched into a new square area. In [23], a circular area is drawn from a square area, and then two circular areas from two individuals are exchanged. In addition, two new mutation operators are presented. The former inserts a randomly generated variable behind a random position of an individual, and the latter removes randomly chosen variables from an individual [17]. Note, however, that this kind of approach faces variable-dimension search spaces, which will guide the search in a messy way since the search spaces with different dimensions have different optimal solutions.
- *Transforming variable-length individuals into fixed-length individuals using auxiliary variables:* In this kind of approach, the length of each individual in the population is fixed, but a set of auxiliary variables is introduced to control the activation of original variables. In general, the auxiliary variables can be divided into two categories: continuous auxiliary variables and binary auxiliary variables. The original variables are activated when the values of the corresponding continuous auxiliary variables are greater than a predefined threshold or the values of the corresponding binary auxiliary variables are equal to 1. This kind of approach has been applied to many fields such as network planning [24], [25], data clustering [26], architecture optimization of deep convolutional neural networks [27], the vehicle routing problem [28], and satellite orbit reconfiguration [29]. However, the introduction of auxiliary variables increases the length of individuals. As a result, this kind of approach may suffer from the curse of dimensionality.

[1]In the n-point crossover operator, the locations of crossover points in two individuals are generally the same [17].

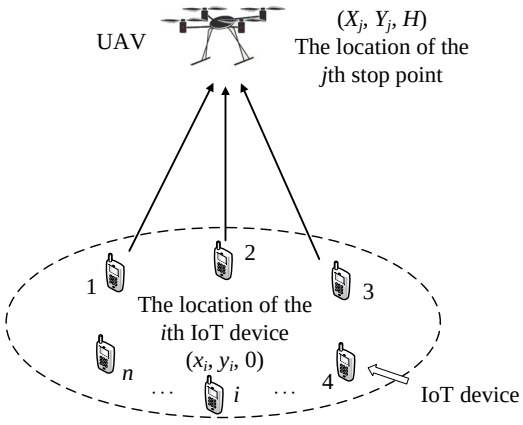


Fig. 1. A UAV-assisted IoT data collection system with a rotary-wing UAV and a set of n ground IoT devices. In this scenario, the 1st, 2nd, and 3rd IoT devices send data to the UAV at the jth stop point.

To avoid the aforementioned issues, we propose a DE algorithm with a variable population size, called DEVIPS, for optimizing the UAV's deployment. In DEVIPS, we design a new encoding mechanism, in which an individual in the population represents the location of a stop point, rather than an entire deployment as in existing approaches. As a result, the whole population represents an entire deployment and the population size is equal to the number of stop points. The main contributions of this work are summarized as follows:

- A UAV-assisted IoT data collection system is studied in this paper. In this system, we optimize the number and locations of stop points simultaneously for the UAV's deployment, which can serve more IoT devices and achieve more energy-effective data collection.
- We propose a new algorithm named DEVIPS to optimize the UAV's deployment. Since the length of each individual in the population is fixed, it is free from the use of special mutation and crossover operators. In addition, the length of each individual is reduced to two. Therefore, the population searches for the optimal number and locations of stop points in a two-dimensional search space.
- We design a new strategy to adjust the population size (i.e., the number of stop points) adaptively. Specifically, the number of stop points can be increased, reduced, or kept unchanged according to the performance improvement in each update. By doing this, an important parameter (i.e., the population size) has been eliminated. In addition, DE serves as the search engine to optimize the locations of stop points.
- Extensive experiments have been carried out on seven instances to investigate the effectiveness of DEVIPS. Compared with five other algorithms, DEVIPS shows better performance.

The rest of this paper is organized as follows. Section II introduces the system model and problem formulation of the UAV's deployment. Section III provides a brief introduction of DE. Section IV describes the proposed DEVIPS. The

experimental setup is given in Section V, followed by the experimental results and discussion in Section VI. Finally, Section VII concludes this paper.

II. System Model and Problem Formulation

As shown in Fig. 1, we consider a UAV-assisted IoT data collection system involving a rotary-wing UAV and a set of n ground IoT devices, denoted as $\mathcal{N} = \{1, 2, \ldots, n\}$. The UAV is employed as a flying data collection platform to collect the data from these ground IoT devices. Due to its agility and mobility, the UAV can change the locations of stop points multiple times, leading to greater coverage and lower energy consumption. Herein, we assume that the number of stop points is k and is unknown *a priori*, and then the set of stop points can be denoted as $\mathcal{K} = \{1, 2, \ldots, k\}$.

We consider that the coordinate of the ith ($i \in \mathcal{N}$) IoT device is known and fixed at $(x_i, y_i, 0)$, where x_i and y_i denote the values in the x-axis and y-axis of the location of the ith IoT device, respectively. Furthermore, we assume that the UAV is flying horizontally at a constant altitude H and the location of the jth ($j \in \mathcal{K}$) stop point is represented by (X_j, Y_j, H), where X_j and Y_j denote the values in the x-axis and y-axis of the location of the jth stop point, respectively. Afterward, the distance between the ith IoT device and the jth stop point is expressed as

$$d_{ij} = \sqrt{(X_j - x_i)^2 + (Y_j - y_i)^2 + H^2},\ \forall i \in \mathcal{N}, j \in \mathcal{K}. \tag{1}$$

The association between the ith IoT device and the UAV at the jth stop point is denoted as binary variable a_{ij}. Specifically, a_{ij} is equal to 1 if the ith IoT device sends data to the UAV at the jth stop point; otherwise, a_{ij} is equal to 0. To save transmission energy, each IoT device always chooses the nearest stop point to send data; thus, a_{ij} is given by

$$\mathcal{C}1 : a_{ij} = \begin{cases} 1, & \text{if } j = \arg\min\limits_{j \in \mathcal{K}} d_{ij}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Also, one has

$$\mathcal{C}2 : \sum_{j=1}^{k} a_{ij} = 1,\ \forall i \in \mathcal{N}. \tag{3}$$

which means that each IoT device chooses only one stop point to send its data.

In addition, considering the system bandwidth limitation, the UAV at each stop point can accept at most M IoT devices to send data simultaneously. Thus, one can obtain

$$\mathcal{C}3 : \sum_{i=1}^{n} a_{ij} \leq M,\ \forall j \in \mathcal{K}. \tag{4}$$

In order to ensure that all IoT devices can be serviced, the following condition should be satisfied:

$$\mathcal{C}4 : \sum_{i=1}^{n} \sum_{j=1}^{k} a_{ij} = n. \tag{5}$$

In our model, the channel gain between the UAV at the jth stop point and the ith IoT device is given by [30], [31],

$$h_{ij} = h_0 d_{ij}^{-2} = \frac{h_0}{(X_j - x_i)^2 + (Y_j - y_i)^2 + H^2}, \quad \forall i \in \mathcal{N}, j \in \mathcal{K}. \tag{6}$$

Therefore, if the ith IoT device sends data to the UAV at the jth stop point, the data rate is given by [32]

$$\begin{aligned} r_{ij} &= B\log_2\left(1 + \frac{p_i h_{ij}}{\sigma^2}\right), \\ &= B\log_2\left(1 + \frac{p_i h_0}{\sigma^2((X_j - x_i)^2 + (Y_j - y_i)^2 + H^2)}\right), \\ &\forall i \in \mathcal{N}, j \in \mathcal{K}, \end{aligned} \tag{7}$$

where p_i is the transmitting power from the ith IoT device to the UAV; h_0 denotes the channel power gain at the reference distance $d_0 = 1$m; σ^2 is the white Gaussian noise power; and B is the system bandwidth.

We assume that the ith IoT device has D_i amount of data sent to the UAV. The time to send the data from the ith IoT device to the UAV at the jth stop point is given by [33], [34]

$$T_{ij} = \frac{D_i}{r_{ij}},\ \forall i \in \mathcal{N}, j \in \mathcal{K}, \tag{8}$$

and the energy consumption is computed as

$$E_{ij} = p_i T_{ij} = \frac{p_i D_i}{r_{ij}},\ \forall i \in \mathcal{N}, j \in \mathcal{K}. \tag{9}$$

Then, the energy consumption of all IoT devices can be given by

$$E_{iot} = \sum_{i=1}^{n} \sum_{j=1}^{k} a_{ij} E_{ij}. \tag{10}$$

Actually, the UAV will hover at each stop point for some time. It will not move to another stop point until all data sent from IoT devices to this stop point have been collected. Therefore, the hover time of the UAV at the jth stop point is given by

$$T_j^h = \max_{i \in \mathcal{N}} \{a_{ij} T_{ij}\},\ \forall j \in \mathcal{K}. \tag{11}$$

Furthermore, the hover energy consumption of the UAV at the jth stop point is expressed as

$$E_j^h = p^h T_j^h,\ \forall j \in \mathcal{K}, \tag{12}$$

where p^h denotes the hover power of the UAV.

Then, the whole energy consumption of the UAV can be written as

$$E_{uav} = \sum_{j=1}^{k} E_j^h. \tag{13}$$

Herein, we ignore the flight energy consumption of the UAV.

The energy consumption of the system is composed of the energy consumption of the UAV and all IoT devices. Thus,

this problem can be formulated as

$$\begin{aligned}
&\min_{\{X_j,Y_j\},k} \quad E_{uav} + \phi E_{iot} \\
\text{s.t.} \quad &\mathcal{C}1: \ a_{ij} \in \{0,1\}, \ \forall i \in \mathcal{N}, j \in \mathcal{K}, \\
&\mathcal{C}2: \ \sum_{j=1}^{k} a_{ij} = 1, \ \forall i \in \mathcal{N}, \\
&\mathcal{C}3: \ \sum_{i=1}^{n} a_{ij} \leq M, \ \forall j \in \mathcal{K}, \\
&\mathcal{C}4: \ \sum_{i=1}^{n}\sum_{j=1}^{k} a_{ij} = n, \\
&\mathcal{C}5: \ X_{min} \leq X_j \leq X_{max}, \ \forall j \in \mathcal{K}, \\
&\mathcal{C}6: \ Y_{min} \leq Y_j \leq Y_{max}, \ \forall j \in \mathcal{K}, \\
&\mathcal{C}7: \ k_{min} \leq k \leq k_{max},
\end{aligned} \tag{14}$$

where $\phi \geq 0$ is the weight between the energy consumption of the UAV and that of all IoT devices; X_{min} and X_{max} are the lower and upper bounds of X_j, respectively; Y_{min} and Y_{max} are the lower and upper bounds of Y_j, respectively; and k_{min} and k_{max} are the lower and upper bounds of k, respectively. Since the UAV serves at most M IoT devices and at least one IoT device at each stop point, k_{min} and k_{max} are equal to $\lfloor \frac{n}{M} \rfloor$ and n, respectively, where $\lfloor \cdot \rfloor$ denotes the rounding down operator.

In this paper, our goal is to optimize the UAV's deployment, including the number and locations of stop points of the UAV, to achieve the minimum energy consumption of the system under all constraints. Specifically, we aim to optimize k and the corresponding $(X_1, Y_1, \ldots, X_k, Y_k)$ to achieve minimum $(E_{uav} + \phi E_{iot})$ while satisfying $\mathcal{C}1 - \mathcal{C}7$. It is clear that $(X_1, Y_1, \ldots, X_k, Y_k)$ represents an entire deployment.

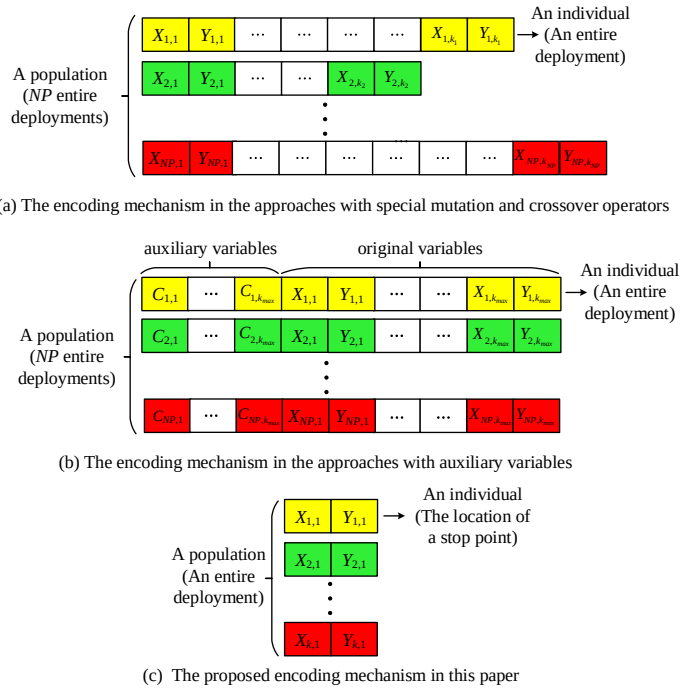
Remark: In this paper, the flight altitude of the UAV is considered as a constant. Next, we would like to give a remark about the flight altitude of the UAV. According to (7), the transmission rate decreases as the flight altitude of the UAV increases; therefore, the transmission time and energy consumption of IoT devices increases accordingly. In contrast, if the flight altitude of the UAV decreases, the probability of the line-of-sight links becomes smaller, which results in a decrease in the transmission rate and an increase in the transmission time and energy consumption of IoT devices. The flight altitude of the UAV will be optimized in our future work.

III. Differential Evolution (DE)

DE, proposed by Storn and Price [35], is a class of simple yet efficient EAs. DE has been applied to solve a variety of optimization problems from diverse fields, such as electromagnetic design [36], order scheduling [37], uniform design [38], and big data optimization [39]. First, DE randomly generates an initial population, in which each individual is called a target vector:

$$\mathcal{P} = \{\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D}), i = 1, 2, \ldots, NP\}, \tag{15}$$

$$x_{i,j} = x_{min,j} + rand \cdot (x_{max,j} - x_{min,j}), \ j \in \{1, 2, \ldots, D\}, \tag{16}$$



(a) The encoding mechanism in the approaches with special mutation and crossover operators

(b) The encoding mechanism in the approaches with auxiliary variables

(c) The proposed encoding mechanism in this paper

Fig. 2. Encoding mechanism in the approaches with special mutation and crossover operators, encoding mechanism in the approaches with auxiliary variables, and the proposed encoding mechanism in this paper, where $X_{i,j}$ and $Y_{i,j}$ denote the values in the x-axis and y-axis of the location of the jth stop point in the ith individual, respectively; $C_{i,j}$ denotes the jth auxiliary variable in the ith individual; and k_{max} is the maximum number of stop points as defined in (14).

where NP is the population size; D is the number of variables; $x_{min,j}$ and $x_{max,j}$ represent the lower and upper bounds of the jth variable, respectively; and $rand$ is a random number uniformly distributed over $[0, 1]$.

After initialization, a mutation operator is executed on each target vector $\mathbf{x}_i$ to generate a mutant vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,D}), \ i \in \{1, 2, \ldots, NP\}$. Then, a trial vector $\mathbf{u}_i = (u_{i,1}, u_{i,2}, \ldots, u_{i,D}), \ i \in \{1, 2, \ldots, NP\}$ is obtained by implementing a crossover operator on $\mathbf{v}_i$ and $\mathbf{x}_i$. Finally, a selection operator chooses the better one between $\mathbf{u}_i$ and $\mathbf{x}_i$ to survive into the next generation.

Next, we introduce the mutation, crossover, and selection operators [40], [41].

1) ***Mutation***: The commonly used mutation operators in the literature are

- DE/rand/1

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}), \tag{17}$$

- DE/rand/2

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) + F \cdot (\mathbf{x}_{r4} - \mathbf{x}_{r5}), \tag{18}$$

- DE/current-to-rand/1

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}), \tag{19}$$

- DE/current-to-best/1

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}), \tag{20}$$

where $r1$, $r2$, $r3$, $r4$, and $r5$ are five distinct integers randomly selected from $[1, NP]$ and are also different from i; $\mathbf{x}_{best}$ denotes the best target vector in the current population; and F is the scaling factor.

2) ***Crossover***: The binomial crossover is given as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand_j \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{21}$$

where j_{rand} is a random integer selected from $[1, D]$ to make sure that $\mathbf{u}_i$ is different from $\mathbf{x}_i$ in at least one dimension; $rand_j$ denotes a uniformly distributed random number over $[0, 1]$ for each j, and CR is the crossover control parameter.

3) ***Selection***: For a minimization problem, the selection operator is implemented as:

$$\mathbf{x}_i = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \leq f(\mathbf{x}_i), \\ \mathbf{x}_i, & \text{otherwise.} \end{cases} \tag{22}$$

IV. Proposed Approach

A. Motivation

In this subsection, we first analyze the drawbacks of the existing approaches introduced in Section I for the UAV's deployment optimization.

- For the approaches with special mutation and crossover operators [18], [23], as shown in Fig. 2(a)[2], each individual in the population of size NP represents an entire deployment, which consists of the locations of a set of stop points. Then the population represents NP entire deployments. In this kind of approach, the length of each individual is not fixed. For example, the lengths of the three individuals with k_1, k_2, and k_{NP} stop points in Fig. 2(a) are $2k_1$, $2k_2$, and $2k_{NP}$, respectively. As already stated, since the length of each individual in the population is not fixed, the special mutation and crossover operators need to be adopted in this kind of approach [17]. It is clear that the dimension of the search space may change from generation to generation for each individual, thus causing confused search.
- As shown in Fig. 2(b), with respect to the approaches with auxiliary variables [24], the individual and the population are also encoded into an entire deployment and NP entire deployments, respectively. In this kind of approach, the length of each individual in the population is $3k_{max}$, consisting of k_{max} auxiliary variables (i.e., $C_{i,1}, \ldots, C_{i,k_{max}}, i \in \{1, 2, \ldots, NP\}$) and $2k_{max}$ original variables (i.e., $X_{i,1}, Y_{i,1}, \ldots, X_{i,k_{max}}, Y_{i,k_{max}}$). Note that k_{max} is defined in (14). The auxiliary variables are used to control the activation of the corresponding original variables. Specifically, $C_{i,j}$ controls the activation of the original variables $X_{i,j}$ and $Y_{i,j}$. Due to the fact that each individual has the same length, the traditional fixed-length mutation and crossover operators can be used directly. However, the length of each individual will drastically increase due to the introduction of auxiliary variables, thus leading to a high-dimensional search space.

It is interesting to observe that although the number of stop points that the UAV needs to visit is variable during the evolution, the dimension of the location of each stop point is fixed. Since the flying height H of the UAV is a constant in this paper, the dimension of the location of each stop point can be considered as two (i.e., the x-axis and y-axis). Moreover, the values of the locations of all stop points in the x-axis and y-axis have the same lower and upper bounds as introduced in (14) (i.e., X_{min} and X_{max} in the x-axis, and Y_{min} and Y_{max} in the y-axis, respectively). Under this condition, if each individual represents the location of a stop point, the length of each individual in the population is the same (i.e., two). More importantly, the auxiliary variables are unnecessary.

Inspired by this observation, we have designed a new encoding mechanism as shown in Fig. 2(c): the location of each stop point is encoded into an individual; thus, the whole population represents an entire deployment and the population size is equal to the number of stop points. The advantages of the proposed encoding mechanism over other encoding mechanisms can be summarized as follows:

- For the encoding mechanisms in both the approaches with special mutation and crossover operators and the approaches with auxiliary variables, the population includes NP entire deployments. Note that NP is a preset parameter and cannot be changed during the evolution. As pointed out in [42], NP has a significant effect on the performance of an algorithm. In contrast, in the proposed encoding mechanism, the population size is equal to the number of stop points, which is optimized during the evolution. Therefore, this parameter has been eliminated.
- Regarding the encoding mechanism in the approaches with special mutation and crossover operators, according to $\mathcal{C}7$ in (14), the number of stop points ranges from k_{min} to k_{max}; therefore, the length of each individual in the population varies from $2k_{min}$ to $2k_{max}$. The variable-length individuals pose significant challenges to EAs [17]. However, the length of each individual in the proposed encoding mechanism is always fixed. In this manner, the special mutation and crossover operators are not required.
- With respect to the encoding mechanism in the approaches with auxiliary variables, the length of each individual in the population is $3k_{max}$. But for the proposed encoding mechanism, the length of each individual in the population is equal to two. Consequently, the dimension of the search space can be remarkably reduced.

Based on the proposed encoding mechanism, a new algorithm named DEVIPS is proposed to optimize the UAV's deployment.

B. DEVIPS

1) General Framework: The general framework of DEVIPS is presented in **Algorithm 1**. First, an initial population $\mathcal{P}$ is randomly generated in the search space. During

[2]Note that the value of the flying height H of the UAV is a constant in this paper. Therefore, we only show the values in the x-axis and y-axis of the location of each stop point.

Algorithm 1 General Framework of DEVIPS

```
FEs = 0; // FEs denotes the number of fitness evaluations
Initialize P according to Algorithm 2;
while FEs ≤ MaxFEs do
    Q = ∅;
    for i = 1 to |P| do
        Implement "DE/rand/1" in (17) and the binomial crossover in (21) on x_i in P to generate u_i;
        Q = Q ⋃ u_i;
    end for
    Update P according to Algorithm 3;
end while
```

Algorithm 2 Initialization of $\mathcal{P}$

```
Randomly generate the locations of k_max stop points, which form an initial P and represent an initial UAV deployment;
Check the feasibility of P via the constraints in (14);
FEs = FEs + 1; // since P presents an entire deployment, the number of fitness evaluations for checking the feasibility of P is 1.
while P is infeasible and FEs ≤ MaxFEs do
    Regenerate the locations of k_max stop points randomly, which form another initial P and represents another initial UAV deployment;
    Check the feasibility of P via the constraints in (14);
    FEs = FEs + 1; // since P presents an entire deployment, the number of fitness evaluations for checking the feasibility of P is 1.
end while
return P
```

Algorithm 3 Updating of $\mathcal{P}$

```
for i = 1 to |Q| do
    P_1 ← insert the ith individual in Q to P;
    P_2 ← utilize the ith individual in Q to replace a randomly selected individual in P;
    P_3 ← delete a randomly selected individual in P;
    Check the feasibilities of P_1, P_2, and P_3;
    if there exists at least one feasible population among P_1, P_2, and P_3 then
        Evaluate the fitness value(s) of the feasible population(s);
        if there exists performance improvement against P then
            the feasible population with the greatest performance improvement is used to replace P;
        else
            if the feasible fitness value of P_3 is equal to that of P then
                P_3 is used to replace P;
            end if
        end if
    end if
    FEs = FEs + 3; // since each of P_1, P_2, and P_3 represents an entire deployment, the number of fitness evaluations for checking the feasibility and evaluating the fitness value of each of them is one.
end for
return P
```

the evolution, DEVIPS performs the mutation and crossover operators of DE to produce the offspring population $\mathcal{Q}$. In this paper, "DE/rand/1" in (17) and the binomial crossover in (21) are employed. Afterward, $\mathcal{P}$ is updated for the next generation. This procedure continues until the stopping criterion is met (i.e., the maximum number of fitness evaluations $MaxFEs$ is reached). In the following, the initialization of $\mathcal{P}$ and the updating of $\mathcal{P}$ in DEVIPS are introduced in detail.

2) Initialization of $\mathcal{P}$: In the initialization, the locations of k_{max} ($k_{max} = n$ as introduced in Section II) stop points[3] are randomly generated, which form an initial population $\mathcal{P}$. Note that $\mathcal{P}$ represents an initial UAV deployment. Then, we check if $\mathcal{P}$ is feasible (i.e., if all constraints in (14) are satisfied). If $\mathcal{P}$ is feasible, the initial UAV deployment is successfully generated; otherwise, the locations of k_{max} stop points are regenerated and the feasibility of $\mathcal{P}$ is rechecked until $\mathcal{P}$ is feasible or $FEs \geq MaxFEs$. **Algorithm 2** describes the initialization of $\mathcal{P}$. Note that $\mathcal{P}$ presents an entire deployment; thus, the number of fitness evaluations for checking the feasibility of $\mathcal{P}$ is one.

3) Updating of $\mathcal{P}$: In DEVIPS, the population size of $\mathcal{P}$ is equal to the number of stop points. In order to optimize the number of stop points, the population size of $\mathcal{P}$ should be variable. In this paper, we consider that the following two aspects are crucial toward the change of the population size:

- The population size can be increased, reduced, or kept unchanged since we do not know the optimal number of stop points.
- The population size should not be changed drastically since drastic changes may lead to an unstable search.

In this paper, we propose a simple and adaptive strategy to accomplish these two aspects. Our main idea is to update at most one individual in $\mathcal{P}$ in each update, which not only changes the population size dynamically, but also maintains a stable search.

After the offspring population $\mathcal{Q}$ is generated, the first individual in $\mathcal{Q}$ is used to generate two new populations:

- It is incorporated into $\mathcal{P}$ to generate a new population $\mathcal{P}_1$.
- It replaces a randomly selected individual in $\mathcal{P}$ to produce another new population $\mathcal{P}_2$.

In addition, a third new population $\mathcal{P}_3$ is produced after an individual randomly selected from $\mathcal{P}$ is removed.

Subsequently, the feasibilities of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are checked. If there exists at least one feasible population among them, the fitness value(s) of the feasible population(s) is/are evaluated. Then, the feasible population with the greatest performance improvement against $\mathcal{P}$ is used to replace $\mathcal{P}$. Note that if no performance improvement occurs but the feasible fitness value of $\mathcal{P}_3$ is equal to that of $\mathcal{P}$, then $\mathcal{P}$ will be replaced by $\mathcal{P}_3$. It means that the individual removed from $\mathcal{P}$ has no effect on the performance; therefore, this individual is redundant. The remaining individuals in $\mathcal{Q}$ experience this process one by one. **Algorithm 3** describes the updating of $\mathcal{P}$. Note that each of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ represents an entire deployment; thus, the number of fitness evaluations for checking the feasibility and evaluating the fitness value of each of them is one. Moreover, in the updating of $\mathcal{P}$, the selection operator adopted in this paper is different from the original selection operator in (22) since our operator selects a better population rather than a better individual.

The population size of $\mathcal{P}$ is denoted as $|\mathcal{P}|$. Then, the population sizes of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are $(|\mathcal{P}| + 1)$, $|\mathcal{P}|$, and $(|\mathcal{P}| - 1)$, respectively. When $\mathcal{P}_1$, $\mathcal{P}_2$, or $\mathcal{P}_3$ is selected to update $\mathcal{P}$, the population size of $\mathcal{P}$ is increased, kept unchanged, or reduced, respectively, which results in an adaptive change of the population size. In fact, only one individual (i.e., the location of one stop point) in $\mathcal{P}$ is different from $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, which suggests that at most one individual

[3]Since a greater number of stop points make it easier to form a feasible population, the locations of k_{max} stop points are generated.

(i.e., the location of one stop point) is updated in each loop of **Algorithm 3**. Therefore, $\mathcal{P}$ is updated in a stable way. In addition, since the feasible population with the greatest performance improvement among $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ is selected to update $\mathcal{P}$, it is beneficial to accelerate the convergence.

C. Discussion

In the following, we discuss the major characteristics of DEVIPS.

- This paper aims at optimizing the number and locations of stop points in the UAV's deployment jointly. In principle, DEVIPS achieves the optimization of the number of stop points by adaptively updating the population size. In addition, DEVIPS optimizes the locations of stop points by DE.
- DEVIPS treats the whole population as an entire deployment (i.e., a solution) and changes the location of one stop point at most in each update. Therefore, DEVIPS is similar to the local search. It is worth noting that the local search typically changes the location of one stop point randomly or based on the locations of nearby stop points. However, in DEVIPS, the location of the new stop point is generated by the crossover and mutation operators of DE, which makes it possible to utilize the information of the locations of all other stop points. As a result, DEVIPS has a better global search capability than the local search. But the global search capability of DEVIPS might still be limited due to the fact that a single deployment is optimized during the evolution. Fortunately, the search capability of DEVIPS can meet our requirements since the search space is only two-dimensional, which has been verified in the experimental studies later.
- DEVIPS has the following advantages: 1) its implementation is simple; 2) it does not introduce any additional parameters; 3) it eliminates an important parameter (i.e., the population size); 4) it searches for the optimal number and locations of stop points in a two-dimensional search space; and 5) it does not add any complex operators and its computational complexity is the same as the classical DE.

V. Experimental Setup

This section introduces the experimental setup for investigating the performance of DEVIPS. First, we briefly describe five algorithms used for comparison. Then, the parameter settings in our experimental studies are provided.

A. Algorithms for Comparison

In order to evaluate the performance of DEVIPS, the following five algorithms were considered to be the peer algorithms.

- VLGA [18] adopts a commonly used crossover operator (i.e., uniform crossover) and a special crossover operator (i.e., cut-and-splice crossover) to produce offspring.
- fGA [23] works in the phenotypic space, in which two circular areas with the same size are selected from two square areas for exchange, and then two new square areas are generated.
- JGGA [24] introduces auxiliary variables to control the expression of original variables. In addition, for improving search ability, a jumping-gene transposition strategy is designed.
- JADE [43] implements the mutation operator "DE/current-to-$p$best" with optional archive and tunes parameters adaptively. Note that in JADE, since the number of stop points cannot be changed during the evolution, it should be preset.
- DEEM [44] uses the encoding mechanism similar to DEVIPS, where the population, rather than an individual, represents an entire deployment. Like JADE, when addressing the UAV's deployment, it also requires a preset number of stop points.

Overall, VLGA and fGA are two approaches with special mutation and crossover operators, JGGA is an approach with auxiliary variables, and JADE and DEEM are two approaches with a preset number of stop points. Therefore, by comparing DEVIPS with these approaches, we can provide multi-facet performance comparisons.

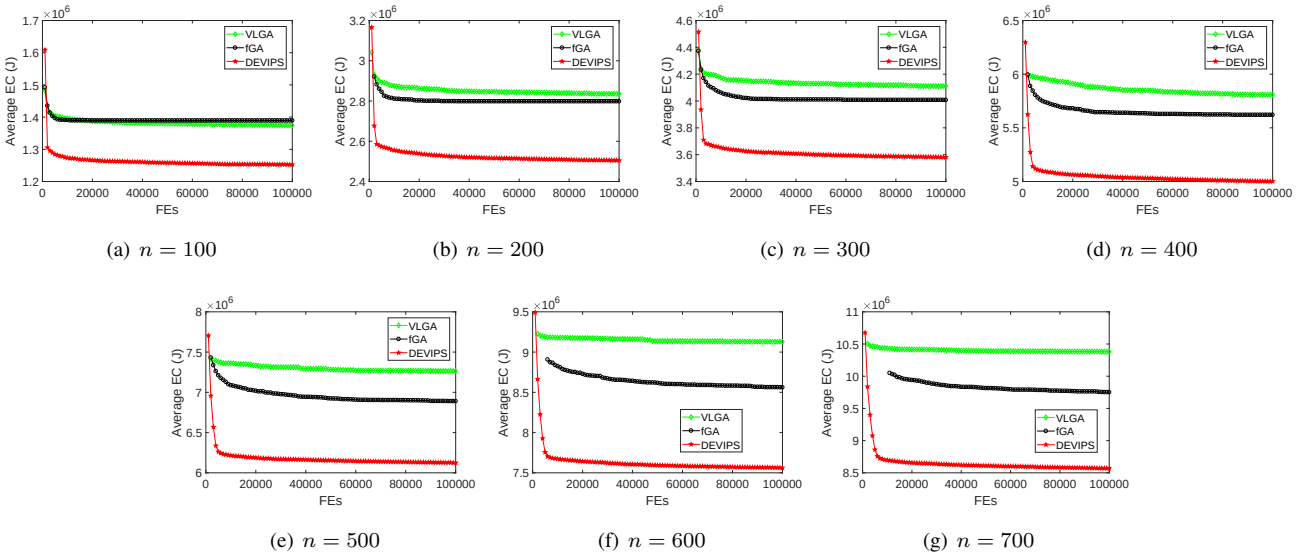
TABLE I
Parameter Settings of the Six Compared Algorithms.

Algorithm	Parameter Settings
VLGA	$NP = 100, p_c = 0.9, p_m = 1/individual\_length$
fGA	$NP = 100, p_c = 0.9, p_m = 0.1$
JGGA	$NP = 100, p_c = 0.8, p_m = 0.04, p_j = 0.01, T_r = 3$
JADE	$NP = 100$
DEEM	$F = 0.9, CR = 0.9$
DEVIPS	$F = 0.6, CR = 0.5$

B. Parameter Settings

The parameter settings of the five peer algorithms and DEVIPS are presented in Table I. Note that the parameter settings of the five peer algorithms were kept the same as in their original papers.

- Population size: The population size of DEEM was the same with the preset number of stop points. The population size of DEVIPS was adaptively varied during the evolution. For other algorithms, the population size was set to 100.
- Parameters for crossover and mutation: For VLGA, fGA, and JGGA, the crossover probability p_c was set to 0.9, 0.9, and 0.8, respectively, and the mutation probability p_m was set to the reciprocal of the individual length, 0.1, and 0.04, respectively. For DEEM and DEVIPS, the scaling factor F was set to 0.9 and 0.6, respectively, and the crossover control parameter CR was set to 0.9 and 0.5, respectively. For JADE, F and CR were updated in an adaptive manner.
- Specific parameters: For JGGA, the jumping probability p_j was set to 0.01 and the number of transposons T_r was set to 3.
- Stopping criterion and number of independent runs: Each algorithm terminated at 100,000 fitness evolutions (i.e., $MaxFEs = 100,000$). Each algorithm was independently run 30 times for each instance.



(a) $n = 100$ (b) $n = 200$ (c) $n = 300$ (d) $n = 400$

(e) $n = 500$ (f) $n = 600$ (g) $n = 700$

Fig. 3. Evolution of the average EC (J) obtained by DEVIPS and two approaches with special mutation and crossover operators (VLGA and fGA).

- Significance test: To test the statistical significance between DEVIPS and each of the peer algorithms, the Wilcoxon rank sum test at a 0.05 significance level was carried out. In the experimental results, we used "+", "−", and "≈" to indicate that DEVIPS performed better than, worse than, and similar to its competitor, respectively.

The parameters of the UAV-assisted IoT data collection system were set as follows. We assumed that all IoT devices were randomly distributed in a 1000m×1000m square area and the flight altitude of the UAV was 200m. Additionally, $D_i(i \in \mathcal{N})$ was randomly distributed within [1, 10^3]MB, M was set to 5, $p_i(i \in \mathcal{N})$ was 0.1W, and h_0 and σ^2 were set to -30dB and -250dBm, respectively. In addition, B was set to 1MHz, p^h was set to 1000W, and ϕ was set to 10000. Seven instances with different numbers of IoT devices were used to evaluate the performance of DEVIPS: $n = \{100, 200, 300, 400, 500, 600, 700\}$.

VI. Experimental Results and Discussion

A. Comparison with Two Approaches with Special Mutation and Crossover Operators (VLGA and fGA)

First, we compared the performance of DEVIPS with that of two approaches with special mutation and crossover operators: VLGA and fGA. The results are reported in Table II. In Table II, "Mean" and "Std Dev" indicate the average and standard deviation of the energy consumption (EC) of the UAV-assisted IoT data collection system over 30 runs, and percentages in the square brackets indicate the performance improvement of DEVIPS against VLGA and fGA. In addition, the statistical test results between DEVIPS and each of VLGA and fGA are summarized at the bottom of Table II.

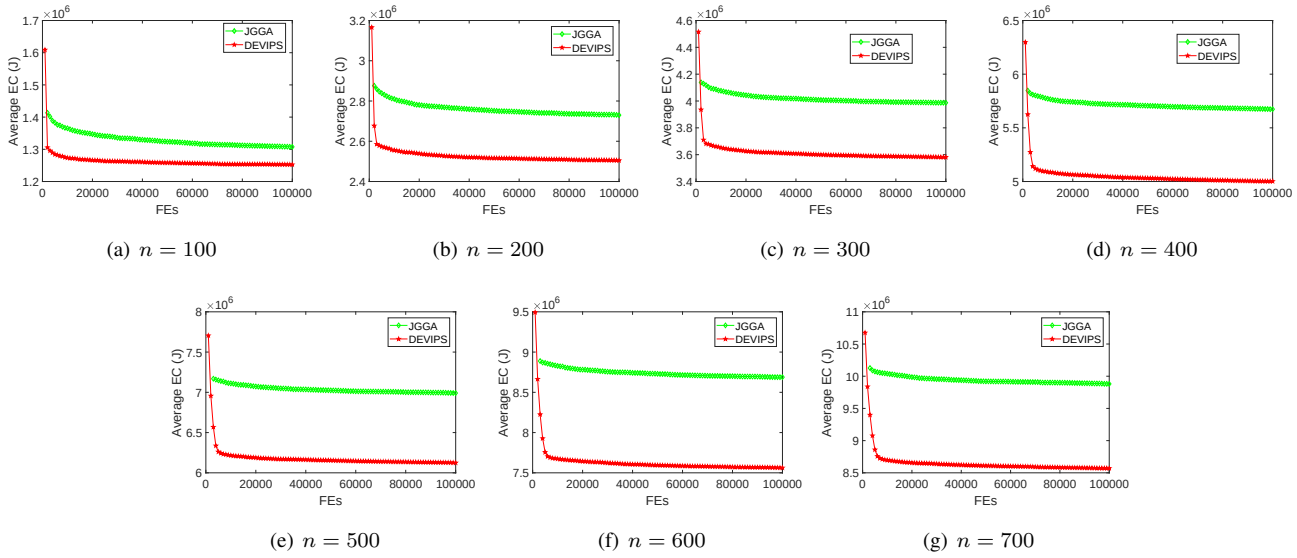
From Table II, it can be seen that DEVIPS is better than VLGA and fGA on each instance in terms of the average EC. By observing the performance improvement, we can see that DEVIPS provides an increasing advantage over other algorithms as the number of IoT devices grows. Specifically, when the number of IoT devices is 100, the performance improvement of DEVIPS is 8.88% and 9.84% against VLGA and fGA, respectively. When the number of IoT devices increases to 700, the performance improvement of DEVIPS against VLGA and fGA reaches 17.44% and 12.09%, respectively. In addition, DEVIPS shows significantly better statistical test results on all instances.

TABLE II
Experimental Results of DEVIPS and Two Approaches with Special Mutation and Crossover Operators (VLGA and fGA).

n	VLGA Mean(Std Dev) (J)	fGA Mean(Std Dev) (J)	DEVIPS Mean(Std Dev) (J)
100	1.3745E+6(6.9440E+3)+ [8.88%]	1.3892E+6(1.3665E+4)+ [9.84%]	1.2525E+6(6.7254E+3)
200	2.8353E+6(1.6711E+4)+ [11.67%]	2.7981E+6(3.0339E+4)+ [10.49%]	2.5045E+6(7.1329E+3)
300	4.1110E+6(3.4598E+4)+ [12.89%]	4.0074E+6(3.1828E+4)+ [10.64%]	3.5809E+6(1.4834E+4)
400	5.8096E+6(5.9486E+4)+ [13.91%]	5.6204E+6(4.2352E+4)+ [11.01%]	5.0016E+6(1.8278E+4)
500	7.2640E+6(9.7987E+4)+ [15.68%]	6.8893E+6(4.1229E+4)+ [11.10%]	6.1248E+6(1.8445E+4)
600	9.1286E+6(8.6089E+4)+ [17.15%]	8.5612E+6(5.2589E+4)+ [11.66%]	7.5628E+6(2.0203E+4)
700	1.0381E+7(4.1259E+4)+ [17.44%]	9.7484E+6(5.7165E+4)+ [12.09%]	8.5702E+6(3.5668E+4)
+	7	7	
−	0	0	
≈	0	0	

This phenomenon can be explained as follows. For VLGA and fGA, due to the fact that the population searches for the optimal solution in the search space with variable dimensions, the changing range of the dimension of the search space drastically expands as the number of IoT devices increases. For example, when $n = 700$, the dimension of the search space ranges from 280 to 1400. In contrast, because the dimension of the search space is always fixed and very low in DEVIPS, it has the capability of obtaining better results.

Fig. 3 presents the evolution of the average EC obtained by



(a) $n = 100$ (b) $n = 200$ (c) $n = 300$ (d) $n = 400$

(e) $n = 500$ (f) $n = 600$ (g) $n = 700$

Fig. 4. Evolution of the average EC (J) obtained by DEVIPS and an approach with auxiliary variables (JGGA).

DEVIPS, VLGA, and fGA. It is clear that DEVIPS converges faster than VLGA and fGA in the early stage and maintains the best performance during the evolution. Note that, when the number of IoT devices is large, Fig. 3 does not show the average EC provided by VLGA and fGA in the early stage. This is because VLGA and fGA cannot produce any feasible solution under this condition. In the initialization, for VLGA and fGA, the number of stop points in each individual is randomly selected between k_{min} and k_{max} (i.e., $\lfloor \frac{n}{M} \rfloor$ and n), which is less than that of DEVIPS (DEVIPS randomly generates the locations of k_{max} stop points in the initialization). Therefore, it is very possible that VLGA and fGA cannot provide enough number of stop points for IoT devices, thus leading to infeasible solutions.

B. Comparison with an Approach with Auxiliary Variables (JGGA)

Subsequently, DEVIPS was compared with an approach with auxiliary variables: JGGA. Table III presents the average and standard deviation of EC over 30 runs, as well as the performance improvement and the statistical test results between DEVIPS and JGGA.

From Table III, DEVIPS provides better average EC and is statistically better than JGGA on each instance. In terms of the performance improvement, the superiority of DEVIPS against JGGA increases with an increase in the number of IoT devices. To be specific, when the number of IoT devices is 100, the performance improvement of DEVIPS is 4.21%. When the number of IoT devices reaches 700, DEVIPS achieves 13.26% performance improvement. The reason is straightforward: the dimension of the search space of DEVIPS is significantly smaller than that of JGGA. For example, when the number of IoT devices increases to 700, the dimension of the search space of JGGA is up to 2100, compared with two in DEVIPS.

Fig. 4 plots the evolution of the average EC derived from DEVIPS and JGGA. Similar to Fig. 3, DEVIPS has faster convergence speed in the early stage and maintains better performance during the evolution.

TABLE III
EXPERIMENTAL RESULTS OF DEVIPS AND AN APPROACH WITH AUXILIARY VARIABLES (JGGA).

n	JGGA Mean(Std Dev) (J)	DEVIPS Mean(Std Dev) (J)
100	1.3075E+6(7.9167E+3)+ [4.21%]	1.2525E+6(6.7254E+3)
200	2.7298E+6(1.0189E+4)+ [8.25%]	2.5045E+6(7.1329E+3)
300	3.9862E+6(1.0855E+4)+ [10.17%]	3.5809E+6(1.4834E+4)
400	5.6737E+6(1.5983E+4)+ [11.85%]	5.0016E+6(1.8278E+4)
500	6.9905E+6(1.8433E+4)+ [12.38%]	6.1248E+6(1.8445E+4)
600	8.6865E+6(2.6654E+4)+ [12.94%]	7.5628E+6(2.0203E+4)
700	9.8807E+6(2.8966E+4)+ [13.26%]	8.5702E+6(3.5668E+4)
+ / – / ≈	7 / 0 / 0	

C. Comparison with Two Approaches with a Preset Number of Stop Points (JADE and DEEM)

We compared DEVIPS with two approaches with a preset number of stop points: JADE and DEEM. For JADE and DEEM, we tested three fixed numbers of stop points: k_{min}, k_{max}, and $(k_{min} + k_{max})/2$ (i.e., $\lfloor \frac{n}{M} \rfloor$, n, and $\lfloor \frac{(M+1)n}{2M} \rfloor$). The number of stop points in DEVIPS was variable during the optimization process as previously mentioned. Table IV gives the average and standard deviation of EC derived from the three compared algorithms over 30 runs. In addition, the performance improvement and the statistical test results of DEVIPS against each of JADE and DEEM are presented. Note that, if some IoT devices could not be served in one run, the run is considered to be infeasible. Under this condition, we only give the feasible rate in Table IV.

As shown in Table IV, JADE with k_{min} stop points and

TABLE IV
EXPERIMENTAL RESULTS OF DEVIPS AND TWO APPROACHES WITH A PRESET NUMBER OF STOP POINTS (JADE AND DEEM).

n	JADE Mean(Std Dev) (J)			DEEM Mean(Std Dev) (J)			DEVIPS Mean(Std Dev) (J)
	k_{min}	$(k_{min}+k_{max})/2$	k_{max}	k_{min}	$(k_{min}+k_{max})/2$	k_{max}	
100	0%+	1.4043E+6(8.1711E+3)+ [10.81%]	1.4837E+6(6.5274E+3)+ [15.58%]	0%+	1.2917E+6(1.2002E+4)+ [3.03%]	1.3507E+6(1.6116E+4)+ [7.27%]	1.2525E+6(6.7254E+3)
200	0%+	2.8508E+6(1.2649E+4)+ [12.15%]	2.9912E+6(1.4947E+4)+ [16.27%]	0%+	2.5698E+6(1.6770E+4)+ [2.54%]	2.7035E+6(2.2978E+4)+ [7.36%]	2.5045E+6(7.1329E+3)
300	0%+	4.1088E+6(1.5469E+4)+ [12.85%]	4.3166E+6(1.3839E+4)+ [17.04%]	0%+	3.6821E+6(2.1229E+4)+ [2.75%]	3.8755E+6(2.7542E+4)+ [7.60%]	3.5809E+6(1.4834E+4)
400	0%+	5.7990E+6(2.2978E+4)+ [13.75%]	6.0787E+6(1.6834E+4)+ [17.72%]	0%+	5.1451E+6(2.7631E+4)+ [2.79%]	5.3782E+6(3.0406E+4)+ [7.00%]	5.0016E+6(1.8278E+4)
500	0%+	7.1617E+6(3.5852E+4)+ [14.48%]	7.4686E+6(1.4416E+4)+ [17.99%]	0%+	6.2840E+6(2.9284E+4)+ [2.53%]	6.5991E+6(3.4363E+4)+ [7.19%]	6.1248E+6(1.8445E+4)
600	0%+	33%+	9.2240E+6(1.9429E+4)+ [18.01%]	0%+	30%+	8.0769E+6(5.5218E+4)+ [6.37%]	7.5628E+6(2.0203E+4)
700	0%+	3%+	1.0434E+7(2.5772E+4)+ [17.86%]	0%+	0%+	9.1732E+6(4.0661E+4)+ [6.57%]	8.5702E+6(3.5668E+4)
+	7	7	7	7	7	7	
−	0	0	0	0	0	0	
≈	0	0	0	0	0	0	

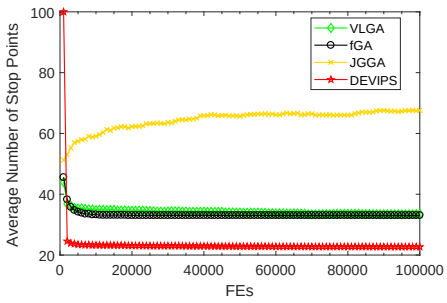


Fig. 5. Evolution of the average number of stop points provided by VLGA, fGA, JGGA, and DEVIPS over 30 independent runs when $n = 100$.

TABLE V
EXPERIMENTAL RESULTS OF DEVIPS AND GAVIPS.

n	GAVIPS Mean(Std Dev) (J)	DEVIPS Mean(Std Dev) (J)
100	1.2550E+6(7.2343E+3)≈	1.2525E+6(6.7254E+3)
200	2.5073E+6(9.8594E+3)≈	2.5045E+6(7.1329E+3)
300	3.5937E+6(1.0907E+4)≈	3.5809E+6(1.4834E+4)
400	5.0103E+6(1.1551E+4)≈	5.0016E+6(1.8278E+4)
500	6.1314E+6(2.0702E+4)≈	6.1248E+6(1.8445E+4)
600	7.5768E+6(2.6715E+4)≈	7.5628E+6(2.0203E+4)
700	8.5899E+6(2.2860E+4)≈	8.5702E+6(3.5668E+4)
+ / − / ≈	0 / 0 / 7	

DEEM with k_{min} stop points failed to produce any feasible run. In addition, in the case of $n = 100$, 200, 300, 400, and 500, the average EC resulting from JADE with k_{max} stop points and DEEM with k_{max} stop points is consistently worse than that provided by JADE with $(k_{min} + k_{max})/2$ stop points and DEEM with $(k_{min} + k_{max})/2$ stop points, respectively. However, when $n = 600$ and 700, both JADE with $(k_{min}+k_{max})/2$ stop points and DEEM with $(k_{min}+k_{max})/2$ stop points cannot provide a 100% feasible rate. The poor performance of JADE and DEEM with a fixed number of stop points signifies that it is not a good choice to set a fixed number of stop points in advance. It is clear that DEVIPS outperforms JADE and DEEM on all instances according to the Wilcoxon's rank sum test at a 0.05 significance level. Moreover, DEVIPS achieves a 100% feasible rate on all instances. The superior performance of DEVIPS not only suggests that optimizing the number and locations of stop points simultaneously can serve more IoT devices and achieve cost-effective data collection, but also demonstrates the effectiveness of the adaptive adjustment of stop points.

D. Evolution of the Average Number of Stop Points

In this subsection, we investigated the evolution of the average number of stop points of different algorithms. We only considered the algorithms without a preset number of stop points (i.e., VLGA, fGA, JGGA, and DEVIPS). Fig. 5 depicts the evolution of the average number of stop points provided by VLGA, fGA, JGGA, and DEVIPS over 30 independent runs in the case of $n = 100$. Note that for VLGA, fGA, and JGGA, in each run, we recorded the number of stop points of the best individual in the population during the evolution. From Fig. 5, it is clear that, overall, the average numbers of stop points of VLGA, fGA, and DEVIPS first decrease and then remain unchanged. Moreover, DEVIPS can quickly obtain a smaller average number of stop points. Different from the previous three algorithms, the average number of stop points in JGGA increases over the course of evolution. This is because JGGA cannot eliminate redundant stop points during the evolution.

E. Effect of the Search Engine

In this subsection, we investigated the effect of the search engine on the performance of DEVIPS. We employed a commonly used search engine, i.e., GA, to replace DE used in DEVIPS. The resultant variant was named GAVIPS. It is worth noting that particle swarm optimization (PSO) needs to use the best individual in the population, but there is no explicit definition of the best individual in our encoding mechanism. Therefore, PSO was not considered. As shown in Table V, there is no significant performance difference between DEVIPS and GAVIPS. This is largely because the search space is two-dimensional and both DE and GA can effectively explore such a low-dimensional search space.

F. Effect of the Update Strategy

To study the effect of the update strategy, we tested DEVIPS with six different update strategies in the case of $n = 100$. Specifically, in each update, at most one, three, five, seven,

TABLE VI
EXPERIMENTAL RESULTS OF DEVIPS WITH SIX DIFFERENT UPDATE STRATEGIES IN THE CASE OF $n = 100$.

Update Strategy	Mean(Std Dev) (J)
1	1.2525E+6(6.7254E+3)
2	1.2731E+6(6.6975E+3)+
3	1.2841E+6(6.3535E+3)+
4	1.2894E+6(1.5840E+4)+
5	1.2981E+6(1.0731E+4)+
6	1.3171E+6(1.4000E+4)+

nine, and 11 individuals were changed in the population. The average EC of each update strategy over 30 runs is presented in Table VI. As shown in Table VI, the update strategy that changes at most one individual in each update can provide the lowest EC. In addition, this update strategy is significantly better than the five competitors according to the Wilcoxon's rank sum test at a 0.05 significance level. The comparison reveals that drastic changes of the population size may lead to performance degradation; thus, DEVIPS only changes at most one individual in each update.

VII. CONCLUSION

In this paper, a DE algorithm with a variable population size, named DEVIPS, was proposed to optimize the number and locations of stop points for the UAV's deployment in a UAV-assisted IoT data collection system simultaneously. The main characteristics of DEVIPS can be summarized as follows:

1) First, we proposed a new encoding mechanism, in which the location of each stop point is treated as an individual, meaning the whole population represents an entire deployment.
2) After that, the dimension of the search space is equal to two, and the population size is equal to the number of stop points. This paper transformed the optimization of the number of stop points into the adjustment of the population size. Then, we proposed a strategy to adaptively adjust the population size.
3) DE was utilized to optimize the locations of stop points in a two-dimensional search space.

DEVIPS was applied to a set of instances with different numbers of IoT devices and compared with two approaches with special mutation and crossover operators, and an approach with auxiliary variables. The experimental results demonstrated that DEVIPS had the best performance. Subsequently, DEVIPS was compared with two approaches with a preset number of stop points. The experimental results not only indicated that optimizing the number and locations of stop points simultaneously is beneficial for improving system performance, but also verified the effectiveness of the adaptive adjustment of stop points in DEVIPS. In addition, three experiments were conducted to study the evolution of the number of stop points and the effects of the search engine and update strategy.

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

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] W. Wang and M. Zhang, "Tensor deep learning model for heterogeneous data fusion in Internet of Things," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018, in press, DOI:10.1109/TETCI.2018.2876568.

[3] J. A. Guerrero-Ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and Internet of Things technologies," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 122–128, 2015.

[4] E. Mezghani, E. Exposito, and K. Drira, "A model-driven methodology for the design of autonomic and cognitive IoT-based systems: Application to healthcare," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 224–234, 2017.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[6] L. Zheng, S. Chen, S. Xiang, and Y. Hu, "Research of architecture and application of Internet of Things for smart grid," in *Computer Science & Service System (CSSS), 2012 International Conference on*. IEEE, 2012, pp. 938–941.

[7] Y. Kawamoto, H. Nishiyama, Z. M. Fadlullah, and N. Kato, "Effective data collection via satellite-routed sensor system (SRSS) to realize global-scaled Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3645–3654, 2013.

[8] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile Internet of Things: Can UAVs provide an energy-efficient mobile architecture?" in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[9] ——, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7574–7589, 2017.

[10] D.-T. Ho, E. I. Grøtli, P. Sujit, T. A. Johansen, and J. B. Sousa, "Optimization of wireless sensor network and UAV data acquisition," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 1, pp. 159–179, 2015.

[11] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, 2019.

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

[13] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[14] M. Alzenad, A. El-Keyi, and H. Yanikomeroglu, "3D placement of an unmanned aerial vehicle base station for maximum coverage of users with different QoS requirements," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 38–41, 2018.

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

[16] W. Du, W. Ying, P. Yang, X. Cao, G. Yan, K. Tang, and D. Wu, "Network-based heterogeneous particle swarm optimization and its application in uav communication coverage," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019, in press, DOI: 10.1109/TETCI.2019.2899604.

[17] M. Ryerkerk, R. Averill, K. Deb, and E. Goodman, "A survey of evolutionary algorithms using metameric representations," *Genetic Programming and Evolvable Machines*, pp. 1–38, 2019.

[18] C.-K. Ting, C.-N. Lee, H.-C. Chang, and J.-S. Wu, "Wireless heterogeneous transmitter placement using multiobjective variable-length genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 4, pp. 945–958, 2009.

[19] S. Lee, S. Lee, K. Kim, and Y. H. Kim, "Base station placement algorithm for large-scale LTE heterogeneous networks," *PloS One*, vol. 10, no. 10, pp. 1–19, 2015.

[20] S.-C. Huang, M.-K. Jiau, and C.-H. Lin, "Optimization of the carpool service problem via a fuzzy-controlled genetic algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1698–1712, 2015.

[21] U. Maulik and S. Bandyopadhyay, "Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 1075–1081, 2003.

[22] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer, "Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 189–203, 2003.

[23] Y.-H. Zhang, Y.-J. Gong, T.-L. Gu, Y. Li, and J. Zhang, "Flexible genetic algorithm: A simple and generic approach to node placement problems," *Applied Soft Computing*, vol. 52, pp. 457–470, 2017.

[24] T.-M. Chan, K.-F. Man, K.-S. Tang, and S. Kwong, "A jumping-genes paradigm for optimizing factory WLAN network," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 33–43, 2007.

[25] Y.-J. Gong, M. Shen, J. Zhang, O. Kaynak, W.-N. Chen, and Z.-H. Zhan, "Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 4, pp. 900–912, 2012.

[26] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2008.

[27] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[28] A. O. Adewumi and O. J. Adeleke, "A survey of recent advances in vehicle routing problems," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 1, pp. 155–172, 2018.

[29] Y. Chen, V. Mahalec, Y. Chen, X. Liu, R. He, and K. Sun, "Reconfiguration of satellite orbit for cooperative observation using variable-size multi-objective differential evolution," *European Journal of Operational Research*, vol. 242, no. 1, pp. 10–20, 2015.

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

[31] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328 – 331, 2018.

[32] P. Huang, Y. Wang, K. Wang, and Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Transactions on Cybernetics*, 2019, in press, DOI:10.1109/TCYB.2019.2916728.

[33] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[34] K. Wang, P. Huang, K. Yang, C. Pan, and J. Wang, "Unified offloading decision making and resource allocation in meran," *IEEE Transactions on Vehicular Technology*, 2019, in press, DOI:10.1109/TVT.2019.2926513.

[35] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[36] M. O. Akinsolu, B. Liu, V. Grout, P. I. Lazaridis, M. E. Mognaschi, and P. Di Barba, "A parallel surrogate model assisted evolutionary algorithm for electromagnetic design optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 93–105, 2019, in press, DOI: 10.1109/TETCI.2018.2864747.

[37] W. Du, W. Zhong, Y. Tang, W. Du, and Y. Jin, "High-dimensional robust multi-objective optimization for order scheduling: A decision variable classification approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 293–304, 2019.

[38] Y. Wang, B. Xu, G. Sun, and S. Yang, "A two-phase differential evolution for uniform designs in constrained experimental domains," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 665–680, 2017.

[39] N. R. Sabar, J. Abawajy, and J. Yearwood, "Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 315–327, 2017.

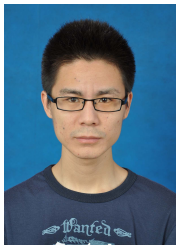
[40] Z.-Z. Liu, Y. Wang, S. Yang, and K. Tang, "An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1403–1416, April 2019.

[41] Y. Wang, B.-C. Wang, H.-X. Li, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2938–2952, 2015.

[42] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[43] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[44] Y. Wang, H. Liu, H. Long, Z. Zhang, and S. Yang, "Differential evolution with a new encoding mechanism for optimizing wind farm layout," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1040–1054, 2018.



Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering both from the Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in control science and engineering, Central South University, Changsha, China. His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.

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

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

Dr. Wang is an Associate Editor for the *Swarm and Evolutionary Computation*. He was a Web of Science highly cited researcher in Computer Science in 2017 and 2018.

Kezhi Wang received the B.E. and M.E. degrees in School of Automation from Chongqing University, China, in 2008 and 2011, respectively. He received the Ph.D. degree in Engineering from the University of Warwick, U.K. in 2015. He was a senior research officer in University of Essex, U.K. Currently he is a Lecturer in Department of Computer and Information Sciences at Northumbria University, U.K. His research interests include wireless communication, mobile edge computing and artificial intelligence.

Kun Yang received the Ph.D. degree from the Department of Electronic & Electrical Engineering of University College London (UCL), UK.

He is currently a Chair Professor in the School of Computer Science and Electronic Engineering, University of Essex, leading the Network Convergence Laboratory (NCL), UK. He is also an affiliated professor at UESTC, China. His main research interests include wireless networks and communications, data and energy integrated networks, and computation-communication cooperation. He manages research projects funded by various sources such as UK EPSRC, EU FP7/H2020 and industries. He has published 150+ journal papers.

He serves on the editorial boards of both IEEE and non-IEEE journals. He is a Senior Member of IEEE (since 2008) and a Fellow of IET (since 2009).