Computer Networks 183 (2020) 107577

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Offspeeding: Optimal energy-efficient flight speed scheduling for UAV-assisted edge computing☆

Weidu Ye, Junzhou Luo *, Feng Shan, Wenjia Wu, Ming Yang

School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

ARTICLE INFO

Keywords:
UAV
Edge computing
Data collection
Energy consumption
Flight speed scheduling
Optimal algorithm

ABSTRACT

Millions of Internet of Thing (IoT) devices have been widely deployed to support applications such as smart city, industrial Internet, and smart transportation. These IoT devices periodically upload their collected data and reconfigure themselves to adapt to the dynamic environment. Both operations are resource consuming for low-end IoT devices. An edge computing enabled unmanned aerial vehicle (UAV) is proposed to fly over to collect data and complete reconfiguration computing tasks from IoT devices. Distinct from most existing work, this paper focuses on flight speed scheduling that allocates proper flight speed to minimize the energy consumption of the UAV with a practical energy model, under the constraints of individual task execution deadlines and communication ranges. We formulate the Energy-Efficient flight Speed Scheduling (*EESS*) problem, and devise a novel diagram to visualize and analyze this problem. An optimal energy-efficient flight speed scheduling (Offspeeding) algorithm is then proposed to solve the offline version of the *EESS* problem. Utilizing Offspeeding and the optimal properties obtained from the theoretical analysis, an online heuristic speed scheduling algorithm is developed for more realistic scenarios, where information from IoT devices keeps unknown until the UAV flies close. Finally, simulation results demonstrate our online heuristic is near optimal. This research sheds light on a new research direction, *e.g.,* deadline driven UAV speed scheduling for edge computing with a practical propulsion energy model.

1. Introduction

Millions of Internet of Thing (IoT) devices have been widely deployed to support applications such as smart city, industrial Internet, and smart transportation. The main task for an IoT device is to collect data and upload these data for centralized analysis, and the other important task is to reconfigure themselves to adapt to the dynamic environment, periodically. However, both operations consume energy. First, long distance wireless data transmission is quite energy consuming; second, reconfiguration involves intensive computing on the newly collected data. Reconfiguration ensures an IoT device to use a matching configuration for the dynamic environment, so it is important to save energy. However, IoT devices are resource limited, usually lack on-board energy and computing power. Therefore, it is critical to avoid long distance wireless data transmission and on-board intensive computing for reconfiguration.

UAV-assisted edge computing is a promising technology, in which a computational edge node is mounted on a UAV, which is dispatched to fly close to each IoT device to perform data collection and reconfiguration computing. As a result, an IoT device waits until the UAV approaches nearby itself and transmits the collected data, so the transmission distance is short. At the same time, intensive computing for reconfiguration no longer performs on the IoT device. Instead, the mounted edge node conducts the computing as soon as it receives all the data, and returns the result to the IoT device before the UAV flies away. In such a way, it reduces the burden of IoT device in respect of energy consumption. Compared to methods based on traditional ground base stations and ground mobile sinks, UAV-assisted edge computing is more flexible due to its high mobility and computing capability, and hence plays an important role in smart city, industrial internet and smart transportation.

Instead of deploying IoT devices in an area, we focus on placing them along a line, which also has lots of applications, such as roads, water/oil/gas pipes or river coast. An illustration of an application scenario for this system is given in Fig. 1. In this scenario, a set of

☆ This work was partially supported by the National Key R&D Program of China (Nos. 2017YFB1003000 and 2018YFB0803400); the National Natural Science Foundation of China (Nos. 61632008, 61532013, 61702097, 62072102 and 62072103); Jiangsu Provincial Key Laboratory of Network and Information Security (No. BM2003201); the Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China (No. 93K-9).

* Corresponding author.

E-mail address: jluo@seu.edu.cn (J. Luo).

https://doi.org/10.1016/j.comnet.2020.107577
Received 4 July 2020; Received in revised form 27 August 2020; Accepted 23 September 2020
Available online 5 October 2020

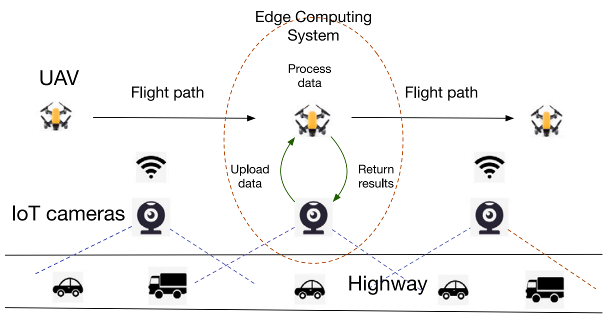


Fig. 1. An illustration of one application scenario. IoT cameras are deployed along a highway to collect traffic data. These IoT cameras are assumed to be capable of self reconfiguration to adapt to the dynamic traffic. However, a good reconfiguration involves intensive computing on recent data and is deadline driven. An edge-computing enabled UAV flies over and collects data from cameras at close positions. Reconfiguration results are returned to these cameras from this UAV. How to control the flight speed to minimize the energy consumption of UAV such that every IoT device has enough time to upload data and each reconfiguration task is completed by its deadline.

IoT surveillance cameras are deployed along a given highway/road to monitor traffic and collect interested data, such as the number of vehicles, types of vehicles, and speeds of them, which can later be used for smart transportation purposes, *e.g.,* road usage and maintenance, and driver behavior analysis. These IoT cameras are capable of self reconfiguration for different traffic situations by changing their sensing interval and sensing quality settings. However, a good reconfiguration depends highly on the most recent data to adapt to the dynamic traffic, *i.e.,* an intensive computing on the recently collected data leads to a good reconfiguration. Therefore, all the reconfiguration computing tasks from IoT devicesare deadline driven. Assume the highway/road is at a remote location where communication infrastructures are lacking, this work uses an edge-computing enabled UAV to collect data from the cameras at close positions and then perform the reconfiguration computing. In this way, energy is saved for the battery powered IoT cameras.

UAV-assisted edge computing is a current research trend in academia [1–17], focusing on offloading computing tasks from IoT devices. Energy is critical for UAVs because of obvious reasons. However, when considering UAV's energy consumption, related works in the literature either ignore propulsion energy [9,10,13,14] or assume a simple flight energy model [1–3,8,15,16]. According to Zeng et al. [18], the propulsion energy may occupy up to 95% total energy consumption of a UAV, a fine-grained propulsion energy model is important to reduced energy consumption of it. Researches have demonstrated that the propulsion power of UAV is mainly affected by its flight speed [5,18–20], hence it is feasible to reduce UAV's propulsion energy consumption via controlling its flight speed. However, simple propulsion energy model is assumed in related works. For example, some related works [2,8] assume the flight power is proportional to the flight speed, and some other related works [3,4] assume the flight power is proportional to the square of flight speed, both are too simple. We adopt a sophisticated UAV flight energy consumption model [19], in which a UAV has a most energy-efficient flight speed, any lower or higher speed consumes more energy.

In UAV-assisted edge computing, execution deadlines for offloaded tasks are important. However, some related works [4,19,21–23] do not take execution time of task into consideration, some other related works [2,5,9,23] assume simple task execution deadlines, *e.g.,* every task shares the same execution deadline. Although the common deadline assumption can ease the theoretical analysis and simplify the problem solving procedure, however, the single deadline model does not explicitly consider execution delay of individual offloaded task, which is critical for IoT devices. We allow individual computing deadlines for each offloaded task and minimize the propulsion energy of UAV by adjusting its flight speed, while considering communication range constraints.

Overall, this paper adopts the most practical flight energy model and allows individual deadline for each offloaded task execution, making our *EESS* problem quite a challenge to be solved optimally. The readers can sense such fundamental nature from the following challenges to our problem. **(1)** According to our practical propulsion energy consumption model, a UAV has a most energy-efficient flight speed, any lower or higher speed consumes more power. It is challenging to choose a speed in case such most energy-efficient flight speed is not feasible subject to other constraints. **(2)** On the one hand, every IoT device must have enough UAV time in its communication range to upload all data, so the UAV should fly slow; on the other hand, each reconfiguration task must be completed by its deadline, so the UAV should fly fast. The best trade-off must be found. **(3)** The communication ranges of IoT devices may overlap each other and the UAV collects data from one device at a time. As a result, every IoT device competes for UAV time to deliver its own data. Moreover, each device has a different amount of data to transmit and a different communication range size, such competition is rather complicated.

The main contributions of this paper are listed as follows:

- A UAV-assisted edge computing system is proposed to collect data and complete offloaded reconfiguration computation from IoT devices. We adopt a sophisticated UAV flight energy consumption model [19], in which a UAV has a most energy-efficient flight speed, any lower or higher speed consumes more power.
- We make the first attempt to minimize propulsion energy of UAV by adjusting its flight speed while considering both communication range constraints and individual computing deadlines for each offloaded task. We then formulate the energy efficient flight speed scheduling (*EESS*) problem.
- A novel diagram is devised to visualize the *EESS* problem in a simple and appealing way. Using the visualization diagram, an optimal energy-efficient flight speed scheduling (Offspeeding) algorithm is proposed, which is theoretically proved to optimally solve the offline *EESS* problem.
- Based on optimality properties obtained from the theoretical analysis for Offspeeding, a heuristic online algorithm is developed for more realistic scenarios, where information of an IoT device keeps unknown until the UAV flies close. Simulation results also show that our online heuristic is near optimal. This research sheds light on a new research direction, deadline driven UAV speed scheduling for edge computing with a practical propulsion energy model.

The rest of the paper is organized as follows. Section 2 introduces works about UAV-assisted edge computing system and energy saving strategies of UAV. Section 3 proposes the system model and formulates the *EESS* problem. A novel graphical virtualization method and some optimality properties are introduced in Section 4. Sections 5 and 6 solve the offline version of *EESS* problem. Section 7 solves the online *EESS* problem. Finally, simulation results are given in Section 8, and conclusions are drawn in Section 9.

2. Related works

In this section, we introduce recent works about UAV-assisted edge computing and UAV's energy saving strategies.

2.1. UAV-assisted edge computing

UAV-assisted edge computing is a promising technology to offload the burden from IoT devices. Yong et al. [18] investigated several works about UAV-assisted wireless network, and believed that UAV mounted edge node is a feasible method to offload IoT devices' task, especially for real-time computing task. Jeong et al. [6] considered

UAV as a mobile cloudlet to assist ground IoT devices completing their computing tasks. They proposed an optimization model that jointly combined UAV's trajectory and offloading strategy together to minimize the energy consumption of UAV and ground users. Hu et al. [7] set the edge-computing enabled UAV to compute the task offloaded from ground users. Different from previous works [6], Hu et al. [7] assumed all tasks uploaded by users were time sensitive, and each task had its own deadline. Cao et al. [8] developed a cellular-connected UAV mobile edge computing systems where the UAV was served by terrestrial base station (TBS) for computation offloading. A resource partitioning strategy was proposed to minimize the UAV energy consumption by jointly optimizing resource partitioning, UAV trajectory and bit allocation. Hu et al. [9] equipped the computation server on UAV to help user equipment (UE) complete their task. A joint optimization problem was formulated to minimize total energy consumption of UAV and UEs under the bandwidth constraints. Zhou et al. [24] also applied the UAV-enabled MEC power system in IoT and WPT (wireless power transfer) technology was used in their work to provide continuous power for UAV to maximize the communication rate.

Above works discuss the UAV-assisted edge computing system in the single-UAV scenario, and there are also a few works focusing on UAV-assisted edge computing system in multi-UAV scenario. Zhang et al. [10] deployed a large swarm of UAVs to enlarge coverage area of IoT devices effectively. UAVs in this work were divided into two layers, where the lower layer were rotor-crafts that flew in lower altitude to serve ground users directly, and the higher layer were fixed-wing UAVs that sent command to lower-layer UAVs. Wang et al. [25] also established a multi-UAV enabled system where a number of UAVs were deployed as flying edge clouds to serve a large scale of users. They proposed a joint optimization problem that considered the number of UAVs and their locations, and decided whether ground users had to offload the task to UAVs.

However, most of these works focus on offloading overwhelming collecting data by the IoT device, and seldom of them concentrating on data reconfiguration of them.

2.2. Energy saving strategies of UAV

Energy saving is another important factor for UAV, for most UAVs are motivated by batteries, and their working time is fundamentally limited by its on-board energy. There are also some works focusing on reducing the energy consumption of UAV. Franco et al. [26] proposed a path planning strategy that covered all the points in a given area by using UAVs. They presented an energy-aware path planning algorithm that minimized consumption while satisfying coverage and resolution constraints. Mazaffari et al. [27] deployed several UAVs in IoT network to collect data from IoT devices. They enabled the uplink reliable communication for IoT devices by minimizing the total transmit energy consumption of UAV. Yong et al. [28] presented an energy consumption model for rotary-wing UAV, including propulsion energy and communication related energy. Based on the model, they minimized the energy consumption of UAV by jointly optimizing UAV's trajectory and network throughput. Alzenad et al. [29] used UAV mounted base station to serve the ground IoT devices in 3-D scenarios, which both considered vertical and horizontal dimensions. They aimed to maximize the number of covered IoT devices by minimizing the transmit power of UAV. Trotta et al. [30] proposed a network architecture where UAVs were dispatched to perform city-scale video monitoring for PoI (Point of interest). To prolong the working time of UAV, they set charging station on the top of bus, therefore the UAV could land on the bus to charge their batteries.

Since propulsion power occupies most power consumption of UAV and is mainly affected by its flight speed, there are also a few works focusing on minimizing propulsion power by optimizing the flight speed of UAV. Yong et al. [19] proposed an energy consumption model for fixed-wings UAV, indicating that flight speed largely affects the propulsion power of UAV. According to this model, they minimized the energy consumption of UAV by finding out the proper flight speed in each time slot. Eom et al. [21] investigated a wireless communication system where the UAV completed tasks transmitted for multiple IoT devices. They aimed to minimize the propulsion power under the constraints of maximum battery storage of ground IoT devices. Xu et al. [22] combined WPT technology to prolong the working time of UAV, while minimizing the propulsion power of it at the same time. They first considered the situation without maximum flight speed constraint, and then added speed constraints into consideration.

However, these works mainly focus on saving UAV's energy by optimizing the air-to-ground communication channel between the UAV and IoT devices, and do not take tasks' computing time into consideration. Besides, these works only achieve the heuristic or approximate solution, but not the optimal result.

2.3. Energy efficient UAV-assisted edge computing

There are also a few related works concentrating on the energy consumption of the UAV and IoT devices [1–5,9]. Yu et al. [1] proposed a novel UAV-enabled edge computing system containing the UAV and edge clouds (ECs) to collaboratively provide MEC serves to IoT devices. They formulated an optimization problem to minimize UAV's energy consumption by considering UAV's location and task allocation. Zhang et al. [2] presented a novel optimization problem that aimed to minimize the total energy consumption including communication energy, computation energy and UAV's flight energy. Zhang et al. [4] proposed a UAV-assisted mobile edge computing system with stochastic computing task model, which aimed to minimize the average weighted energy consumption of sensors and the UAV. Li et al. [5] studied a UAV-assisted mobile edge computing scenario to optimize the computation offloading by minimizing the energy consumption of UAV. Hu et al. [9] aimed to minimize the total energy consumption of the UAV and UEs under the deadline and UAV's trajectory constraints. Liu et al. [23] investigated a UAV-enabled wireless MEC system where an energy transmit server and a MEC server were mounted on UAV. This work aimed to minimize the total required energy by jointly optimizing the CPU frequencies, the offloading amount, transmit power and UAV's trajectory.

However, most of these works have the same deadline for each task, which is not practical in realistic scenario. In this work, we define individual deadline for each task, and attempt to find the minimum energy consumption of UAV.

Overall, most UAV-assisted edge computing systems lack a fine-grained propulsion energy consumption model. Few works focusing on reducing propulsion energy neither take tasks' computing time into consideration or achieve the optimal result. On this basis, we propose an optimal energy-efficient flight speed scheduling algorithm to solve the problem discussed above.

3. System model and problem formulation

3.1. Network model

We consider m IoT devices are deployed along a line to monitor the environment, denoted as $M = \{Sn_1, Sn_2, \cdots, Sn_m\}$, which has lots of applications, such as roads, water/oil/gas pipes or river coast. A UAV flies at a fixed altitude H, and an edge node is mounted on it to collect data and complete the offloaded task from IoT devices. This UAV departs periodically to serve every IoT device, namely a tour. For each UAV tour, the IoT device Sn_i has a reconfiguration task $U_i = (C_i, D_i, T_i)$, where C_i describes the total number of CPU cycles to complete this task; D_i denotes the size of data to be collected; and T_i is the deadline for completing this task, $i = 1, 2, \ldots, m$. The UAV has to return the reconfiguration results back to users before users are uploaded in different solutions. The UAV returns to the airport once it serves all the reconfiguration tasks from IoT devices. During the data collection process, the UAV can collect data from IoT devices when it is flying, but is only allowed to receive data from a single IoT device at each time. While during the task computing process, the UAV is capable of simultaneously executing multiple reconfiguration tasks.

The receiving rate between the UAV and IoT devices is assumed to be R, and CPU frequency of UAV is denoted as f_{UAV}, both are fixed numbers known in advance. On this basis, the receiving time of task i should be:

$$t_i^* = \frac{D_i}{R}$$

where the computing time of task i should be:

$$t_i^+ = \frac{C_i}{f_{UAV}}$$

Since the collected data is large and the CPUs on the UAV are powerful enough, we assume the receiving time t_i^* is generally larger than the computing time t_j^+, *i.e.*, $t_i^* \geq t_j^+, \forall i,j$. Let reconfiguration task deadline be T_i, by which, the UAV should finish data collection and complete reconfiguration task and return the results. We ignore the delay and energy consumption caused by sending reconfiguration settings back to IoT devices, for the data size of reconfiguration results is much smaller compared to the collected raw data.

3.2. Energy consumption model

The energy consumption of a UAV contains three parts: communication energy, computing energy and propulsion energy. Since propulsion energy is much larger than that of computing and communication energy, we only consider the propulsion energy consumption of UAV in this paper.

For a fixed-wing UAV in straight flight with constrained speed, the propulsion energy consumption is expressed in a closed form [19], shown in Eq. (1):

$$p = c_1 * v^3 + \frac{c_2}{v}, \tag{1}$$

where p represents the propulsion power of UAV, c_1 and c_2 are related to the aircraft's weight, wing area, air density, and so on. Therefore, propulsion power of UAV is related to speed v.

The UAV has the optimal flight speed to cost the least energy consumption. On the one hand, it will cost much energy consumption when the flight speed is small enough (the second part of Eq. (1) will be too large). On the other hand, when the flight speed is large enough, it also costs much energy (the first part of Eq. (1) will be too large). Overall, there should be a proper flight speed that the UAV costs the least energy consumption, any higher or lower speed will cost more.

Based on Eq. (1), the optimal speed of UAV to achieve the minimum propulsion energy can be calculated in Appendix F.

3.3. Problem formulation

We assume that communication ranges of IoT devices are overlapped with each other, such scenario is illustrated in Fig. 2. Note that the non-overlapped scenario can be divided into several overlapped scenarios. We assume the UAV flies along this line to collect data and completes offloaded reconfiguration tasks from m IoT devices within their own deadline T_i. The UAV cannot fly at too-fast speed, for it has to finish all the tasks within communication ranges of IoT devices. On the other hand, the UAV cannot fly at too-low speed, for it has to complete the task of each IoT device before the deadline T_i. With the constraints above, we aim to minimize the energy consumption of UAV.

Specifically, we define the UAV's flight speed at position d as $v(d)$, where d is a position in UAV's flight path. Then, the flight time spent for the UAV from $d = 0$ to reach $d = x$ can be calculated by

$$t = \int_0^x \frac{\mathrm{d}d}{v(d)}. \tag{2}$$

IoT device Sn_i is assumed to have a communication range, starting from b_i and terminating at d_i. Sn_i is only allowed to upload its collected data when UAV flies within such range, and the reconfiguration computation results must be returned when the UAV is still inside such range. We assume the communication ranges have different ranges sizes but aligned. In other words, $d_i - b_i$ may vary from IoT device to IoT device, but a range starts left must terminate left and there is no range inside another range. So we have $b_1 < b_2 < \cdots < b_m$ and $d_1 < d_2 < \cdots < d_m$. Without loss of generality, let $b_1 = 0$ and $d_m = D$. Such an assumption is reasonable because the same type of IoT devices usually have similar communication ranges and they are placed in a distance from each other to monitor different areas.

As a result, the UAV visits and serves IoT devices in the order of their indexes. We therefore assume a reasonable and tight deadline for each reconfiguration task, *e.g.*, $T_i = t_i^+ + \sum_{j=1}^{i} t_j^*$. Since the UAV collects data from one IoT device at a time, a total collection time of $\sum_{j=1}^{i-1} t_j^*$ data must have be spent prior to start task i. To finish task i, $t_i^+ + t_i^*$ time is needed, so $t_i^+ + \sum_{j=1}^{i} t_j^*$ is the minimum time to complete task i.

Since for every $Sn_i, i = 1, 2, \ldots, m$, the UAV has to collect data from it, complete the reconfiguration task and return the result to it, hence, at least $t_i^* + t_i^+$ UAV time must be spent within communication range (b_i, d_i). The constraint is written as

$$\int_{b_i}^{d_i} \frac{\mathrm{d}d}{v(d)} \geq t_i^* + t_i^+.$$

However, since communication range of IoT devices are overlapped with each other, flying within the range of Sn_i does not necessarily mean the UAV serves (collects data and completes reconfiguration data) Sn_i. Assume $d = \alpha_i$ is the switching position where the UAV finishes collecting data from Sn_i and starts collecting from Sn_{i+1},and assume $d = \beta_i$ is the returning positions where the UAV returns the reconfiguration result to Sn_i. Let $\alpha_0 = 0$. The switching positions and the returning position must be within the communication range of (b_i, d_i), we therefore have the following *range constraints*.

$$b_i \leq \alpha_{i-1} < \alpha_i < \beta_i \leq d_i, \forall i. \tag{C1}$$

Then, we have the following *completion constraints*

$$\begin{aligned} &\int_{\alpha_{i-1}}^{\alpha_i} \frac{\mathrm{d}d}{v(d)} \geq t_i^*. \\ &\int_{\alpha_i}^{\beta_i} \frac{\mathrm{d}d}{v(d)} \geq t_i^+. \end{aligned} \tag{C2}$$

Since the reconfiguration deadline for Sn_i is defined $T_i = t_i^+ + \sum_0^i t_j^*, i = 1, 2, \ldots, m$. Therefore, the UAV has to complete all tasks from IoT devices Sn_1 to Sn_m before their own deadlines, as the *delay constraint*

$$\int_0^{\beta_i} \frac{\mathrm{d}d}{v(d)} \leq T_i. \tag{C3}$$

According to the propulsion energy model Eq. (1), and since the $v(d)$ is the UAV speed at position d, then we have the flight power $p(d)$ at position d as $p(d) = c_1 \times v(d)^3 + \frac{c_2}{v(d)}, \forall d \in [0, D]$. So the energy consumption at position d is $e = pt = p(d) \times \frac{\mathrm{d}d}{v(d)}$. The total energy consumption of the UAV can be integrated as

$$E = \int_0^D p(d) \frac{\mathrm{d}d}{v(d)} \tag{3}$$

We now define *EESS* problem as follows.

Definition 1 (*EESS Problem*)**.** Given m IoT devices and a UAV-assisted edge computing system described above, the energy efficient flight speed scheduling (*EESS*) problem is to determine the flight speed $v(d)$ for the UAV in each distance point d to minimize the propulsion energy consumption in Eq. (3) while satisfying the *range constraint* Eq. (C1), the *completion constraint* Eq. (C2) and the *delay constraint* Eq. (C3).

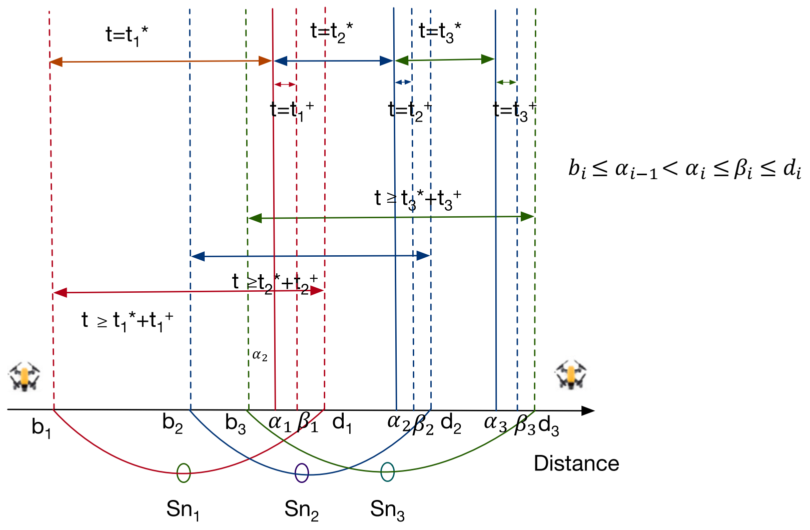


Fig. 2. An illustration of the *EESS* problem. For Sn_i, the UAV spends at least t_i^* time collecting its data, and t_i^+ time computing its task, which has a completion deadline T_i. We assume the UAV collects from one device at a time (e.g., before point α_1, the UAV receives task from Sn_1, and after point α_1, the UAV receives task from Sn_2), but can compute for multiple devices simultaneously, it can also collect and compute simultaneously (e.g. from α_1 to β_1, the UAV computes task from Sn_1 and receives tasks from Sn_2 simultaneously). The UAV's flight energy must be minimized to finish all tasks: on the one hand, if flies slow, the UAV has enough time to collect data however consumes more time and energy according to our practical energy model; on the other hand, a faster-speed flight may satisfy delay constraints and reduce energy consumption but may cause insufficient time in communication range, causing failure of completing tasks (e.g. UAV has to take $t_1^* + t_1^+$ time in Sn_1's communication range from b_1 to d_1). Meanwhile, IoT devices compete for UAV time.

The *EESS* problem is expressed in mathematics model, shown in $P1$.

$$(P1): \min_{v(d)} \quad \int_0^D p(d)\frac{\mathrm{d}d}{v(d)} \tag{4}$$

$$\text{s.t.} \qquad (C1)(C2)(C3) \tag{5}$$

An illustration of the *EESS* problem is given in Fig. 2.

This problem is called offline problem if all information is known before scheduling; it is called online problem if IoT device information is not available unless the UAV flies close. The speed scheduling function is an optimal solution of offline algorithm, and we call it the optimal speed scheduling function, denoted as $v_{opt}(d)$.

4. Graphical virtualization and optimality properties

In this section, we first introduce a novel graphical visualization method to redefine the *EESS* problem, and then present some optimality properties in preparing for the optimal solution.

4.1. Graphical virtualization for the EESS problem

We introduce the graphical virtualization method through a novel distance-time diagram as in Fig. 3. In this diagram, the X-axis represents flight distance and Y-axis indicates the flight time, as a result, any point (d, t) stands for the UAV reaching position d at time t. Therefore, a curve on this diagram is the distance-time accumulation curve, for example curve $L(d)$ in Fig. 3. It is clear that the slope of curve $L(d)$ at d is the reciprocal of UAV's flight speed $\frac{1}{v(d)}$ at position d. Hence, We can determine the optimal speed schedule $v_{opt}(d)$ by determining the optimal accumulation curve $L_{opt}(d)$.

However, a feasible accumulation curve does not go freely on the diagram, it has constraints. According to the *completion constraint* Eq. (C2), the UAV should not fly fast and the curve should not have too small slope, so we have a lower bound for a feasible accumulation curve, $F(d)$. That is before leaving range of Sn_i, a total of $T_i^+ = t_i^+ + \sum_{j=1}^{i} t_j^*$ time must have accumulated at position $d = d_i$. According to the *delay constraint* Eq. (C3), the UAV should not fly slow and the curve should not have too large slope, so we have an upper bound for

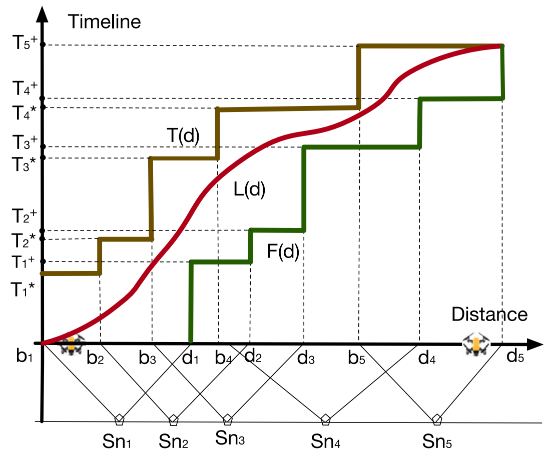


Fig. 3. An illustration of the proposed graphical visualization method. In the distance-time diagram, accumulation curve $L(d)$ specifies the time t when reaching position d, the slope of which is the reciprocal value of UAV's flight speed. Due to the *completion constraint* (*delay constraint*) the UAV should not fly fast (slow), so we have a lower (upper) bound curve $F(d)$ ($T(d)$). A feasible accumulation curve must be within the two bounds. We focus on finding the optimal accumulation curve.

a feasible accumulation curve, $T(d)$. That is before entering range of Sn_i, a total of $T_i^* = \sum_{j=1}^{i-1} t_j^*$ time must have accumulated at position $d = b_i$, for otherwise the deadline can never be met. According to the *range constraint* Eq. (C1), a feasible accumulation curve must be within the two bounds. Our graphical virtualization method is inspired by the data flow model in [31].

4.2. Optimality properties

Before we present the algorithm to optimally solve the *EESS* problem, we would like to first give some optimality properties.

Lemma 1 (*Convexity*). *According to the propulsion energy model in Eq. (1), the energy consumption power for UAV p is a convex function of UAV flight speed v.*

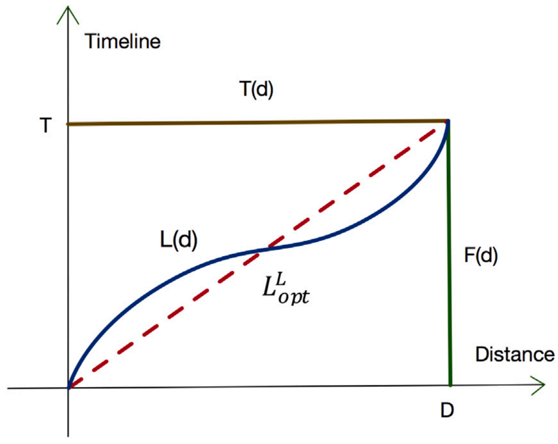


Fig. 4. Using a constant UAV flight speed can achieve a higher energy-efficiency than using a varying flight speed. The optimal accumulation curve is straight between any two points, as long as this is feasible.

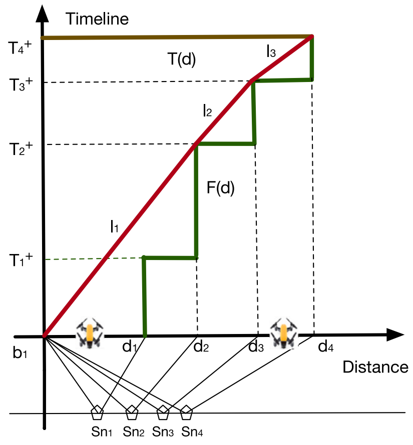


Fig. 5. The graphical visualization of the *EESS-L* problem, where the red sub-lines are the optimal solution. Our algorithm works in iteration. The main steps are described as follows: we connect the origin to every corner one by one, to test the slope of the connected line, and then choose the line with the largest slope as the first sub-line. The problem then repeats.

Proof. See Appendix A.

Lemma 2 (*Constant Speed Property*). *Using a constant UAV flight speed can achieve a higher energy-efficiency than using a varying flight speed.*

Proof. Assume the UAV speed changes. Without loss of generality, let the corresponding accumulation curve be $L(d)$ in Fig. 4. Then, the energy consumption of the UAV is described as:

$$E_{L(d)} = \int_0^D p(v(d)) \frac{\mathrm{d}d}{v(d)} = \int_0^D \frac{p(v(d))}{v(d)} \mathrm{d}d \tag{6}$$

If UAV flies at constant speed $\frac{D}{T}$, energy consumption of it is written as:

$$E^L_{opt} = p(\tfrac{D}{T}) \cdot T = p(\frac{\int_0^D \mathrm{d}d}{\int_0^D \frac{\mathrm{d}d}{v(d)}}) \cdot \int_0^D \frac{\mathrm{d}d}{v(d)} \tag{7}$$

We consider the following version of Jensen's inequation [31],

$$p(\frac{\int_0^D f(d)g(d)\,\mathrm{d}d}{\int_0^D g(d)\,\mathrm{d}d}) \le \frac{\int_0^D p(f(d))g(d)\,\mathrm{d}d}{\int_0^D g(d)\,\mathrm{d}d} \tag{8}$$

Let $f(d) = v(d)$ and $g(d) = \frac{1}{v(d)}$, we get,

$$p(\frac{\int_0^D v(d)\cdot \frac{\mathrm{d}d}{v(d)}}{\int_0^D \frac{\mathrm{d}d}{v(d)}}) \le \frac{\int_0^D p(v(d))\cdot \frac{\mathrm{d}d}{v(d)}}{\int_0^D \frac{\mathrm{d}d}{v(d)}} \tag{9}$$

We multiply T (which is $\int_0^D \frac{1}{v(d)}\mathrm{d}d$) in inequation, and get:

$$p(\frac{\int_0^D v(d)\cdot \frac{\mathrm{d}d}{v(d)}}{\int_0^D \frac{\mathrm{d}d}{v(d)}}) \cdot \int_0^D \frac{\mathrm{d}d}{v(d)} \le \int_0^D p(v(d)) \cdot \frac{\mathrm{d}d}{v(d)} \tag{10}$$

$$p(\frac{\int_0^D \mathrm{d}_d}{\int_0^D \frac{\mathrm{d}d}{v(d)}}) \cdot \int_0^D \frac{\mathrm{d}d}{v(d)} \le \int_0^D p(v(d)) \cdot \frac{\mathrm{d}d}{v(d)} \tag{11}$$

$$p(\tfrac{D}{T}) \cdot T \le \int_0^D p(v(d)) \cdot \frac{\mathrm{d}d}{v(d)} \tag{12}$$

Finally, we have

$$E^L_{opt} \le E_{L(d)} \tag{13}$$

There is a direct corollary that follows Lemma 2.

Corollary 1. *On the distance-time diagram, the optimal accumulation curve is straight between any two points, as long as this is feasible.*

5. Optimal solution for special cases

Since the *EESS* problem is complicated to be solved directly, we start with two simplified special cases. By solving these special cases, we obtain some important properties for the general problem.

5.1. EESS problem sharing starting position

In the *EESS* problem given in Definition 1, when all the IoT devices share a common starting position of their communication ranges, *i.e.,* $b_1 = b_2 = \cdots = b_m = 0$, we call such a special case the *EESS-L* problem. Note that, in the *EESS-L* problem, the communication range terminating positions are different, *i.e.,* $d_1 < d_2 < \cdots < d_m$.

The graphical visualization of *EESS-L* problem is shown in Fig. 5. In which we can see the upper bound curve $T(d)$ is a straight line, where the lower bound curve $F(d)$ stays the same as the general problem. More specially, we have $F(d_i) = t_i^+ + \sum_{j=1}^{i} t_j^*$ and $T(d) = t_m^+ + \sum_{j=1}^{m} t_j^*$. According to Corollary 1, the optimal accumulation curve L_{opt} can be divided into j sub-lines, illustrating as $l_i, i = 1, 2, \ldots, j$ (see as red lines in Fig. 5). Each sub-line l_i starts from position q_i with slope s_i, and terminates at q_{i+1}, and let $q_1 = (0, 0)$.

Before we go directly to the algorithm, we would like to first present some important properties about the optimal accumulation curve L^L_{opt} of the *EESS-L* problem.

Lemma 3. *An optimal accumulation curve L^L_{opt} must only decrease its slope.*

Proof. See Appendix B.

Lemma 4. *An optimal accumulation curve L^L_{opt} only decreases its slope at a lower bound curve corners, i.e., $F(d_i), i = 1, 2, \ldots, m$.*

Proof. See Appendix C.

With the above two lemmas ready, we now introduce the high level idea of our algorithm. It is clear that any optimal accumulation curve L^L_{opt} consists of a set of sub-line l_i such that L^L_{opt} changes direction only by decreasing slope and only at corners of $F(d)$. Therefore, if we find all the direction changing corners, the L^L_{opt} curve can be determined. We therefore focus on finding the very first direction changing corner, *e.g.,* determining the first sub-line. We connect the origin to every corner one by one to test the slope of the connected line. We choose the line with the largest slope as the first sub-line, and the corresponding corner as the first direction changing corner. Starting from this corner, the same problem repeats and we find the next sub-line and corners.

The detailed pseudo code is given in Algorithm 1.

Algorithm 1 Optimal algorithm for *EESS-L*

```
Input:  Receiving time for each IoT device $t_i^*$
        Computing time for each IoT device $t_i^+$
        Communication range of each IoT device $i$, from 0 to $d_i$
        Number of users $m$, UAV's initial state $q_1$
Output: UAV's flight speed $v_j$ in each iteration $j$
        UAV's flying state $q_j$ in each iteration $j$
        UAV's flying energy $E_{fly}$
1: $i = 1$, $j = 1$, $v = \emptyset$, $E_{fly} = 0$
2: while $i <= m$ do
3:     Find the maximum slope sub-line $l_j$ with slope $s_j$ from UAV's current state $q_j$ to every corner of curve $F(d)$ after distance $d_i$
4:     $v_j = \frac{1}{s_j}$
5:     Update $q_{j+1}$ as the terminal point of $l_j$ and update $i$ as the corresponding IoT device.
6:     $j = j + 1$
7: end while
8: Calculating $E_{fly}$ according to $v$ and $q$
9: return $v$, $E_{fly}$, $q$
```

This algorithm works in iteration. In each iteration, one sub-line and its corresponding direction changing corner is determined. Note that variable q_j is defined as the origin flight state for UAV in iteration j. Algorithm 1 attempts to find the maximum-slope line l_j with slope s_j from the origin point q_j to direction changing corner point of F(d) $(d_i, F(d_i))$. Then the optimal flight speed of UAV is $v_j = \frac{1}{s_j}$. Finally, the algorithm updates the origin state of UAV for next iteration q_{j+1} and its corresponding IoT device i, and the algorithm terminates when the UAV serves all the IoT devices $(i > m)$.

Theorem 1. *Algorithm 1 computes the optimal solution for the EESS-L problem in $O(m^2)$ steps.*

Proof. We prove by contradiction. Supposed the solution produced by Algorithm 1 is not optimal, and the optimal accumulation curve L_{opt}^L is different from our computed one. We next show such L_{opt}^L cannot be optimal.

An example is given in Fig. 6, in which, the accumulation curves produced by our algorithm is $l_1, l_2, l_3 \in L^L$. Assuming the optimal accumulation curve L_{opt}^L is different from L^L. There are two cases, either part of L_{opt}^L curve is below L^L curve, or part of L_{opt}^L is above L^L.

Case 1. As shown in Fig. 6, instead of using sub-line l_1 to connect the origin with the second corner as by the L^L curve, the L_{opt}^L curve uses two sub-lines l_1' and l_2' to connect the origin and the second corner, and these two sub-lines are below sub-line l_1. Obviously, we have the sub-line slopes inequation $s_2' > s_1 > s_1'$, where s_2', s_1 and s_1' are the slopes for sub-lines l_2', l_1 and l_1' respectively. According to Lemma 2 and Corollary 1, the energy consumption for solution by L^L curve is smaller than that of L_{opt}^L, which contradicts the assumption that L_{opt}^L is optimal.

Case 2. Assume part of L_{opt}^L curve is above L^L curve, as an example is given in Fig. 6, and assume the slope of curve l_3^{opt} satisfies $s_3^{opt} > s_1$. As shown in Fig. 6, instead of using sub-line l_1, l_2 and l_3 to connect the origin to final destination $F(d_m)$. the L_{opt}^L curve uses two sub-lines l_3' and l_4' to connect the origin and the second corner, and these two sub-lines are above sub-line l_1, l_2 and l_3. Obviously, we can find another straight line l_5' between line l_3' and l_4'. We have the sub-line slopes inequation $s_3' > s_5' > s_4'$, where s_3', s_5' and s_4' are the slopes for sub-lines l_3', l_5' and l_4' respectively. According to Lemma 2 and Corollary 1, the energy consumption for solution by L^L curve is smaller than that of L_{opt}^L, which contradicts the assumption that L_{opt}^L is optimal.

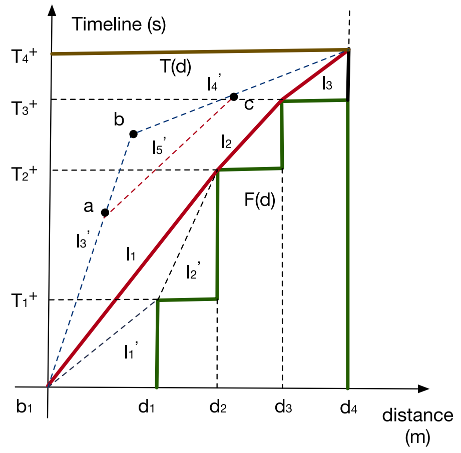


Fig. 6. An illustration of Algorithm 1 results for the *EESS-L* problem, where the accumulation curves produced by Algorithm 1 is $l_1, l_2, l_3 \in L'$ (red solid lines). We prove L' to be optimal by contradiction. In case 1, instead of l_1, L_{opt}^L uses two sub-lines l_1', l_2' instead. Obviously, $s_2' > s_1 > s_1'$, indicating L_{opt}^L is not optimal. In case 2, L_{opt}^L uses l_3', l_4' instead, where slope $s_3' > l_1$. Then, line s_5' could be found between l_3' and l_4' where $s_3' > s_5' > s_4'$, indicating L_{opt}^L is not optimal. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Since both cases are impossible, so we must have $L_{opt}^L = L^L$.

During each iteration, Algorithm 1 attempts to find the maximum slope from the origin state q_j to every direction changing corner of $F(d)$. The maximum steps for Algorithm 1 is $\frac{m*(m-1)}{2}$, whose time complexity is $O(m^2)$.

5.2. EESS problem sharing terminating position

In the *EESS* problem given in Definition 1, when all the IoT devices share a common terminal position of their communication ranges, *i.e.,* $d_1 = d_2 = \cdots = d_m = 0$, we call such a special case the *EESS-U* problem. Note that, in the *EESS-U* problem, the communication range terminating positions are different, *i.e.,* $b_1 < b_2 < \cdots < b_m$.

The graphical visualization of *EESS-U* problem is shown in Fig. 7. In which we can see the lower bound curve $F(d)$ is a straight line, where the upper bound curve $T(d)$ stays the same as the general problem. More specially, we have $T(d_i) = \sum_{j=1}^{i} t_j^*$ and $F(d_i) = d_m$. According to Corollary 1, the optimal accumulation curve L_{opt} can be divided into j sub-lines, illustrating as $l_i, i = 1, 2, \ldots, j$ (see as red lines in Fig. 7). Each sub-line l_i starts from position q_i with slope s_i, and terminates at q_{i+1}, and let $q_1 = (0, 0)$.

Similar to the previous defined *EESS-L* problem, we have two optimality properties about the optimal accumulation curve L_{opt}^U. Their proofs are similar and omitted.

Lemma 5. *An optimal accumulation curve L_{opt}^U must only increase its slope.*

Lemma 6. *An optimal accumulation curve L_{opt}^U only increases its slope at an upper bound curve corner, i.e., $T(d_i), i = 1, 2, \ldots, m$.*

It is clear that L_{opt}^U consists of a set of sub-line l_i such that L_{opt}^U changes direction only by increasing slope and only at corners of $T(d)$. We therefore focus on finding the very first direction changing corner, *e.g.,* determining the first sub-line. We connect the origin to every corner one by one to test the slope of the connected line. We choose the line with the largest slope as the first sub-line, and the corresponding

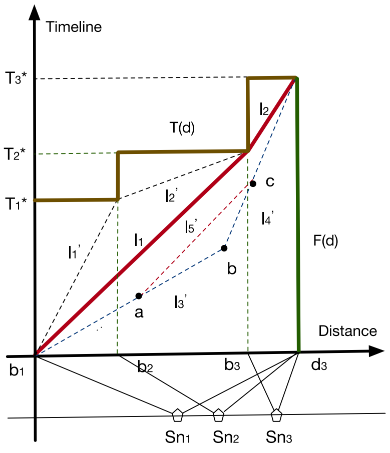


Fig. 7. An illustration of Algorithm 2 results for the *EESS-U* problem, where the accumulation curves produced by Algorithm 1 is $l_1, l_2 \in L'$ (red solid lines). We prove L' to be optimal by contradiction. In case 1, instead of l_1, L^L_{opt} uses two sub-lines l'_1, l'_2 instead. Obviously, $s'_1 > s_1 > s'_2$, indicating L^L_{opt} is not optimal. In case 2, L^L_{opt} uses l'_3, l'_4 instead, where slope $s'_3 < l_1$. Then, line s'_5 could be found between l'_3 and l'_4 where $s'_3 < s'_5 < s'_4$, indicating L^L_{opt} is not optimal.

corner as the first direction changing corner. Starting from this corner, the same problem repeats and we find the next sub-line and corners.

The pseudo code is given in Algorithm 2.

Algorithm 2 Optimal algorithm for *EESS-U*

```
Input:  Receiving time for each IoT device $t^*_i$
        Computing time for each IoT device $t^+_i$
        Communication range of each IoT device $i$, from $(b_i,0)$ to $(d_i,0)$
        Number of IoT devices $m$, UAV's initial state $q_1$
Output: UAV's flight speed $v_j$ in each iteration $j$
        UAV's flying energy $E_{fly}$
1: $i = 1,\ j = 1,\ v = \emptyset,\ E_{fly} = 0$
2: while $i <= m$ do
3:     Find the minimum slope sub-line $l_j$ with slope $s_j$ from UAV's current state $q_j$ to every corner of curve $T(d)$ after distance $b_i$
4:     $v_j = \frac{1}{s_j}$
5:     Update $q_{j+1}$ as the terminal point of $l_j$ and update $i$ as the corresponding IoT device.
6:     $j = j + 1$
7: end while
8: Calculating $E_{fly}$ according to $v$ and $q$
9: return $v$, $E_{fly}$, $q$
```

The correctness of Algorithm 2 is given in the following theorem.

Theorem 2. *Algorithm 2 computes the optimal solution for the EESS-U problem in $O(m^2)$ steps.*

The proof of Theorem 2 is similar to that of Theorem 1, and thus is omitted for brevity.

6. Offspeeding algorithm for the general case

We are now ready to solve the general *EESS* problem, which does not impose any extra restriction on the communication ranges of IoT devices. The optimal accumulation curve changes its direction both by increasing slope and by decreasing slope. However, the optimality properties obtained in the special cases, *i.e.*, Lemmas 4 and 6 still hold. In other words, L_{opt} decreases its slope only at a low bound curve corner, and it increases its slope only at an upper bound curve corner. We therefore present the following lemma directly without proof.

Lemma 7 (*Optimal Accumulation Curve Property*). *An optimal accumulation curve $L_{opt}(d)$ either intersects with $F(d)$ or $T(d)$, i.e., for some i, we have either $L_{opt}(d_i) = F(d_i)$ or $L_{opt}(b_i) = T(b_i)$.*

- *If we have $L_{opt}(d_i) = F(d_i)$ at point d_i, then L_{opt} curve must decrease its slope at $d = d_i$.*
- *If we have $L_{opt}(b_i) = T(b_i)$ at point b_i, then L_{opt} curve must increase its slope at $d = b_i$.*

Lemma 8 (*Uniqueness*). *If there exists an optimal accumulation curve $L_{opt}(d)$ that satisfies Lemma 7, then $L_{opt}(d)$ must be unique.*

Proof. See Appendix D.

Based on Lemmas 7 and 8, we can deduce that if an algorithm obeys optimal accumulation curve property in Lemma 7, the algorithm must be optimal. We therefore propose an optimal energy-efficient flight speed scheduling (Offspeeding) algorithm to solve the offline *EESS* problem. This algorithm guarantees that its output curve obeys the properties in Lemma 7, which is denoted as $L(d)$.

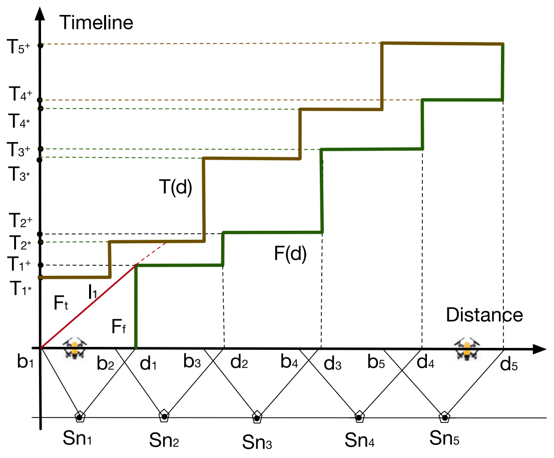
The basic idea is simple: we find the first piece of the sub-line and then the same problem repeats. To find the first sub-line, we only need to locate the first slope changing point. From Lemma 7, we have known that such a point must be either on the lower bound curve $F(d)$ at $d = d_i$ or on the upper bound curve $T(d)$ at $d = b_i$, *i.e.*, the corners of both curves. We therefore test every possible corner. An example is illustrated in Fig. 8. We connect the origin and a corner to see whether the resulting line is feasible. We test every corner in the order of the corner's distance. Unless the current line is infeasible, we continue to test the next corner. The very last feasible sub-line is the one we are looking for, and the corner is the optimal first slope changing point. The detailed formal steps are presented in Algorithm Offspeeding.

Algorithm 3 Offspeeding

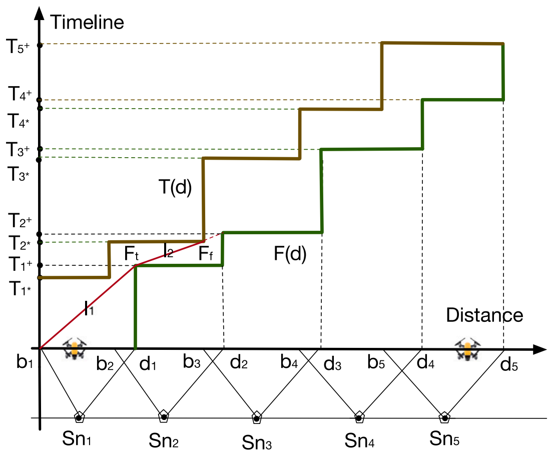
```
Input:  Number of IoT devices $m$
        $(b_i, d_i), t^*_i, t^+_i, i = 1, 2, \cdots, m$
Output: UAV's flight speed $v_j$ in each iteration $j$
        UAV's flying energy $E_{fly}$
1: $i = 1,\ j = 1,\ v = \emptyset,\ E_{fly} = 0,\ q_1 = (0,0)$
2: while $i <= m$ do
3:     $l_j$ is the line from $q_i$ to $(d_i, F(d_i))$
4:     $l'_j$ is the line from $q_i$ to $(b_i, T(b_i))$
5:     $s_j$ is slope of $l_j$
6:     $s'_j$ is slope of $l'_j$
7:     if Extension line of $l_j$ intersects $T(d)$ then
8:         $v_j = \frac{1}{s_j}$
9:         $v = v \cup \{v_j\}$
10:        $j = j + 1,\ q_j = (d_i, F(d_i))$
11:    end if
12:    if Extension line of $l'_j$ intersects $F(d)$ then
13:        $v_j = \frac{1}{s'_j}$
14:        $v = v \cup \{v_j\}$
15:        $j = j + 1,\ q_j = (b_i, T(b_i))$
16:    end if
17:    i=i+1
18: end while
19: Calculate $E_{fly}$ based on $v$ and $q$
20: Return $v$, $E_{fly}$
```

To better understand Algorithm Offspeeding, examples in Fig. 8 are provided to illustrate the detailed steps.

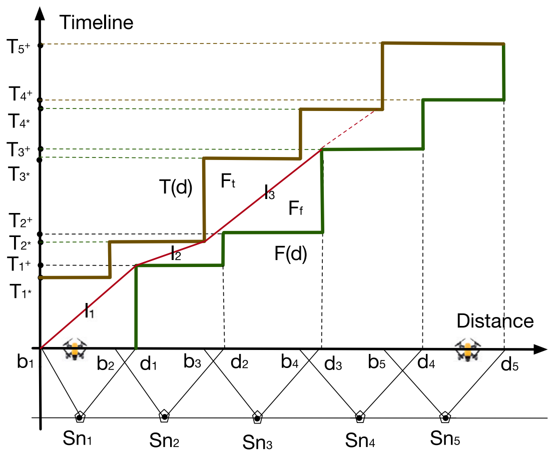
In Fig. 8, the accumulation curve starts from the origin point $(0, 0)$, and it goes between the two bound curve $T(d)$ and $F(d)$. To better



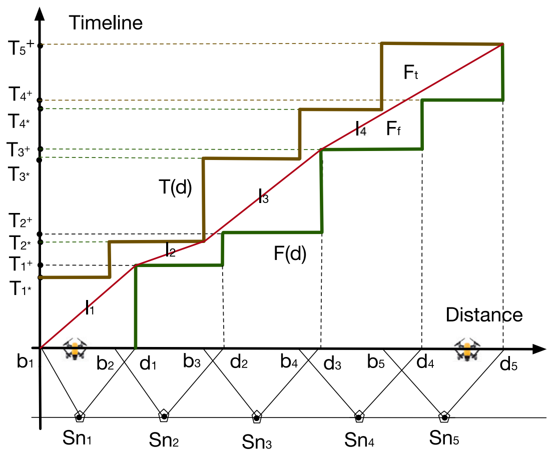
(a) Offspeeding: Step 1



(b) Offspeeding: Step 2



(c) Offspeeding: Step 3



(d) Offspeeding: Step 4

Fig. 8. The way algorithm Offspeeding works. In each step, algorithm Offspeeding attempts to find the longest straight line l_j between $T(d)$ and $F(d)$. Specifically, we test every corner in the order of the corner's value d. Unless the current line is infeasible, we continue to test the next corner. The very last feasible sub-line is the one we are looking for, and the corner is the optimal first slope changing point. The algorithm terminates when the UAV reaches destination d_i, and the slope of each sub-line is the reciprocal number of UAV's velocity.

illustrate algorithm Offspeeding, we define F_t as the set of sub-lines that interact with upper bound $T(d)$, and F_f as set of sub-lines that interact with lower bound $F(d)$. During each step, we attempt to find the boundary sub-line between set F_t and F_f. For instance, in step 1, red-full line l_1 is the boundary sub-line, for l_1 intersects with direction changing corner of $F(d)$ and its extension line intersects with $T(d)$. In step 2, the red-full line l_2 is the boundary sub-line, since l_2 intersects with direction changing corner of $T(d)$ and its extension line intersects with $F(d)$. Step 3 and step 4 are similar to previous steps, and the iteration ends until UAV reaches the destination point d_5.

Theorem 3. *Algorithm Offspeeding can achieve the optimal solution of the EESS problem in $O(m^2)$ step.*

Proof. See Appendix E.

7. The online heuristic algorithm

In the previous section, we propose an optimal offline algorithm Offspeeding to solve the *EESS* problem. In this section, we utilize the results of Offspeeding to consider more realistic scenarios that information from IoT devices keeps unknown until the UAV flies close, and present an online algorithm, called *EDS* (emergency deadline simulation), to minimize the propulsion energy of the UAV.

The basic idea of *EDS* is similar to Algorithm 1, which first connects the UAV's current state q_p to every corner of lower bound $F(d)$ one by one, and chooses the line with largest slope until the UAV flies out of communication range of the mth IoT device. However, different from Algorithm 1, the UAV does not know the existence of IoT devices until flying into their communication range, and can only determine the local optimal flight speed v_j according to the current information from known IoT devices. Therefore, the UAV flight speed obtained by Algorithm *EDS* is a set of flight speed, which is determined at the time when the UAV enters or flies out of the communication range of an IoT device.

In Algorithm *EDS*, parameter v is defined as set of UAV's flight speed, and q_p is the current state of UAV in iteration p (line 1). Variable k represents the number of IoT devices the UAV has already known and j is denoted as the number of IoT devices the UAV has flown passed (line 2).

Once the UAV enters the communication range of a new IoT device (line 4-line 9) or flies out of the communication range of a known IoT device (line 10-line 15), Algorithm *EDS* will determine a new

Algorithm 4 Online algorithm *EDS*

```
Input:  Receiving time for each task $t_i^*$
        Computing time for each task $t_i^+$
        Communication range of each IoT device $i$, from $(b_i,0)$ to $(d_i,0)$
        Number of IoT devices $m$
Output: UAV's flight speed $v$
        UAV's flying energy $E_{fly}$
 1: $q_1 = (0,0)$, $v = \emptyset$
 2: $k = 0$, $j = 1$, $p = 1$
 3: while $j <= m$ do
 4:   if UAV flies into a new IoT device's communication range then
 5:     UAV acquires its current flight state $q_p$
 6:     $k = k + 1$, $m' = k - j + 1$
 7:     Use Algorithm 1 to calculate $v'$ with input of
        $\{m', t_i^*, t_i^+, b_i, d_i, q_p\}$, where $j \le i \le k$.
 8:     $v = v \cup \{v'[1]\}$, $p = p + 1$.
 9:   end if
10:   if UAV flies out of an IoT device's communication range then
11:     UAV acquires its current flight state $q_p$
12:     $j = j + 1$, $m' = k - j + 1$
13:     Use Algorithm 1 to calculate $v'$ with input of
        $\{m', t_i^*, t_i^+, b_i, d_i, q_p\}$, where $j \le i \le k$.
14:     $v = v \cup \{v'[1]\}$, $p = p + 1$.
15:   end if
16: end while
17: Calculate $E_{fly}$ based on $v$ and $q$
18: Return $v$, $E_{fly}$
```

flight speed. Specifically, it first acquires its current state q_p (line 5 and line 11) and the number of current serving tasks m'. Then, we utilize Algorithm 1 to obtain the optimal flight speed v' according to the information of current m' devices (line 7 and line 13). Then the UAV changes its flight speed as the first element of set v', named $v'[1]$, for the following elements in v' might be varied when the UAV flies into communication range of a new IoT device (line 8 and line 14).

Algorithm *EDS* terminates when the UAV flies out of communication range of the mth IoT device and we calculate the total propulsion energy E_{fly} according to set v and q_p (line 17).

Time complexity of Algorithm *EDS* is $O(m^2)$, for the algorithm first travels all the m users and then uses Algorithm 1 to find the optimal flight speed for UAV in IoT device m'. The time complexity of first step is $O(m)$, and the complexity for second step is also $O(m)$. Overall, the time complexity of algorithm *EDS* is $O(m^2)$.

8. Performance evaluation

In this section, simulation experiments are conducted to evaluate the performance of optimal offline algorithm Offspeeding and online algorithm *EDS*.

8.1. Experimental setting

In our experiments, the UAV flies at a fixed altitude $H = 100$ m. A computational edge node is mounted on the UAV to deal with tasks uploaded from 5 to 1000 IoT devices. Each IoT device i only has one individual task u_i to be dealt with, and communication range of IoT devices Cr_i is ranging from 5 m to 60 m. The receiving time t_i^* is within range of 0.005 s to 2 s, and computing time t_i^+ is 0.15 ∼ 0.35 times of t_i^*. Overall, we present simulation parameters, as shown in Table 1[25].

To better illustrate the advantage of our algorithms, we present another online algorithm *LDS* based on a simple strategy. We first define a concept named TPD (Time Per Distance) for each IoT device i, whose value is $\frac{t_i^*+t_i^+}{Cr_i}$ for each i. In algorithm *LDS*, the UAV calculates

Table 1
Simulation parameters.

Parameters	Meaning	Value
t_i^*	Receiving time for each IoT device	0.005 s∼2 s
t_i^+	Computing time for each IoT device	0.15∼0.35 times of t_i^*
m	Number of IoT devices	50∼1000
v_{max}	Maximum flight speed	60 m/s
H	Altitude of the UAV	100 m
c_1	Parameter of energy model	$9.26 * 10^{-4}$
c_2	Parameter of energy model	2250
Cr	Communication range of IoT device	5 m∼60 m

TPD for each IoT device i, and acquires the number of IoT devices it serves each time, denoted as m'. Parameter $Dis(p)$ is defined as the distance where number of IoT devices m' that the UAV serves does not change, and p is number of distance varied from 1 to $2m + 1$. Then, value of TPD in distance $Dis(p)$ is the sum of m' IoT device's TPD number, and the flight speed v is the reciprocal for its TPD. For instance, if the UAV serves two IoT devices i and j in distance $Dis(2)$, then $TPD_{Dis(2)} = TPD_i + TPD_j$ and $V_D = \frac{1}{TPD_D}$, and flight speed $v_{Dis(2)} = \frac{1}{TPD_i+TPD_j}$.

Besides, another proposed algorithm named *ALG* [32] is also used to compare with Algorithm Offspeeding and *EDS*. The core idea of ALG is to divide the transmission task into two sets, the average transmission rate and the remaining rate, and updates the receiving rate of task according to the comparison the flight speed in these two sets, which is a "Max remaining-time" algorithm. If we analog the transmission energy of wireless device as the propulsion energy of UAV, and assume transmission rate as the flight speed of UAV, ALG can be utilized to solve the *EESS* problem proposed in this work.

On the basis, simulation experiments are conducted to evaluate performance of optimal offline algorithm Offspeeding, online algorithm *EDS* and compared algorithms *LDS* and *ALG*.

8.2. Impact of the number of IoT devices

In this part, a group of experiments discussing the relationship between propulsion energy of UAV and the number of IoT devices are conducted. Let $Cr_i = 50$m be communication range for each IoT device i, and the overlapping range between device i and $i+1$ is denoted by x_i, i.e., x_i is 0.15 ∼ 0.35 times of Cr_i. In addition, receiving time for each IoT device is $t_i^* = 1s$ and its computing time t_i^+ is 0.15 ∼ 0.35 times of t_i^*. Based on these settings, we compare the propulsion energy of UAV among Offspeeding, *EDS*, *ALG* and *LDS* while the number of IoT devices is varying from 50 to 1000.

The results are shown in Fig. 9. It can be seen from this figure that Algorithm Offspeeding performs the best and the performance of Algorithm EDS is near to that of Algorithm Offspeeding. Algorithm ALG costs more energy than EDS and Offspeeding, for it only considers the flight speed among two neighboring SNs, EDS considers all the SNs whose information is known by the UAV and Offspeeding considers information of all SNs. The gap between Algorithm Offspeeding and Algorithm EDS is growing when the number of SN increases, for Offspeeding always achieves the optimal results for each SN, but Algorithm EDS does not, leading to a gap between Algorithm Offspeeding and Algorithm EDS. When the number of IoT devices increases, Algorithm EDS will have more chances to work out different solutions than that of Algorithm Offspeeding, leading to a larger accumulative gap between Algorithm EDS and Algorithm Offspeeding.

8.3. Impact of deployment density of IoT devices

In this part, we conduct a group of experiments discussing the UAV's propulsion power and IoT devices' deployment density. Specifically, the total flight distance of UAV is limited as 3000 m, and we deploy

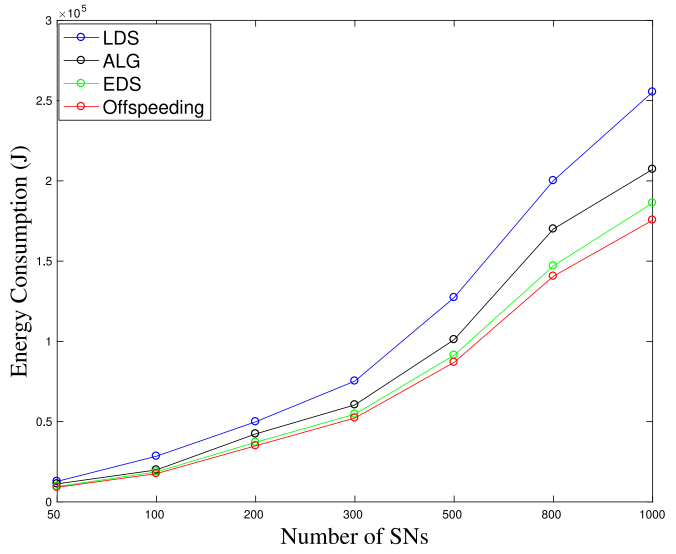


Fig. 9. The impact of IoT device number on algorithm performance by propulsion energy.

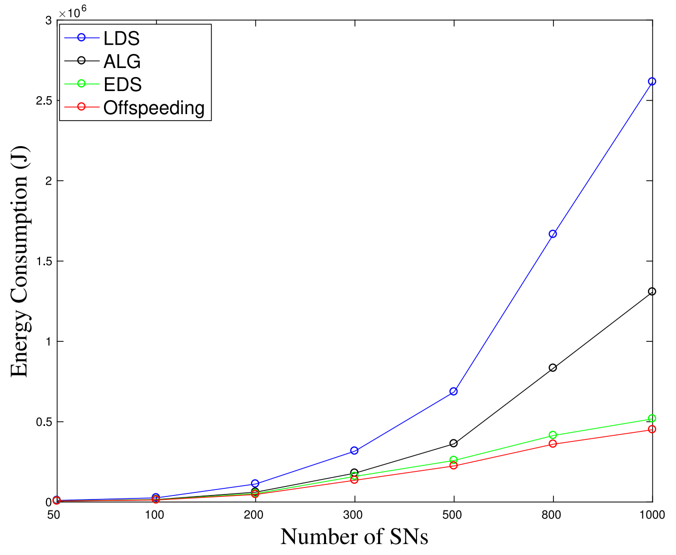


Fig. 10. The impact of the IoT device density on algorithm performance by propulsion energy.

averagely 50 ∼ 1000 IoT devices among the flight path of the UAV. Then, average communication range for each IoT device is $\frac{3000}{m}$.

Parameter $x_i \in X$ is defined as overlapping range between IoT device i and $i+1$, i.e., x_i is 0.15 ∼ 0.35 times of average communication range $\frac{3000}{m}$. To this end, communication range for each IoT device i can be written follows:

$$cr_i = \begin{cases} \frac{3000}{m} + x_i & i = 1 \\ \frac{3000}{m} + x_{i-1} + x_i & 1 < i < m \\ \frac{3000}{m} + x_{i-1} & i = m \end{cases}$$

Receiving time in this experiment is $t_i^* = 1$ s, and computing time t_i^+ is 0.15 ∼ 0.35 times of t_i^*.

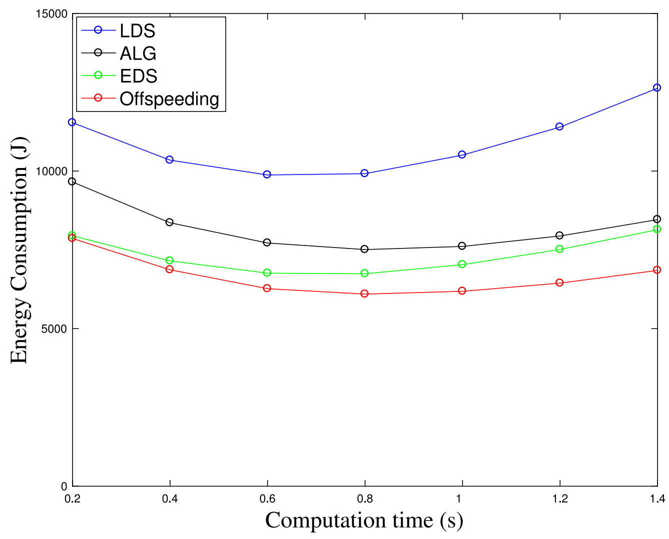


Fig. 11. The impact of the computing time on algorithm performance by propulsion energy.

The results are shown in Fig. 10. Similar to previous experiment, Algorithm Offspeeding achieves the best result in Fig. 10 and result of Algorithm *EDS* is near to that of Algorithm Offspeeding. We can also find that solutions of Algorithm *ALG* and *LDS* consume more energy than that of Algorithm Offspeeding and *EDS*. Moreover, with the increasing number of IoT devices, solutions of all the three algorithms will consume more energy, and the gap between Algorithm *LDS* and Algorithm Offspeeding/*EDS* is also increasing.

8.4. Impact of computing time

In this part, we deploy 500 IoT devices and limit the total flight distance for UAV as 3000 m. Communication range for each IoT device is the same to previous section, and the number of IoT devices is 500. Receiving time for each IoT device is $t_i^* = 5$ s, and computing time t_i^+ ranges from 0.2 s ∼ 1.4 s for this task.

The results are shown in Fig. 11. Algorithm Offspeeding costs the least energy compared with three other algorithms *EDS*, *LDS* and *ALG*. When tasks' computing time t_i^+ increases, energy consumption of the UAV first decreases and reaches the lowest point when $t_i^+ = 0.8$ s. Then, propulsion power of UAV increases when $0.8 \text{ s} < t_i^+ < 1.4$ s. On the one hand, when $t_i^+ < 0.8$ s, computing time for each IoT device is too short, the UAV has to fly faster, which costs more energy than $t_i^+ = 0.8$ s. On the other hand, when $t_i^+ > 0.8$ s, computing time t_i^* is too long, the UAV has to reduce its flight speed and spends more time serving IoT device i, which also costs more energy.

9. Conclusion

We study the energy efficient flight speed scheduling problem for UAV-assisted edge computing in this paper. Distinct from most existing work, this paper focuses on flight speed scheduling that allocates proper flight speed to minimize the energy consumption of the UAV with a practical energy model, under the constraints of individual task execution deadlines and communication ranges. Specifically, we formulate the *EESS* problem and devise a novel diagram to visualize and analyze the problem. Since the *EESS* problem is complicated to be solved directly, two simplified special cases *EESS-L* and *EESS-U* are first analyzed and some important properties are obtained for the general problem. Then, the Algorithm Offspeeding is proposed to solve the

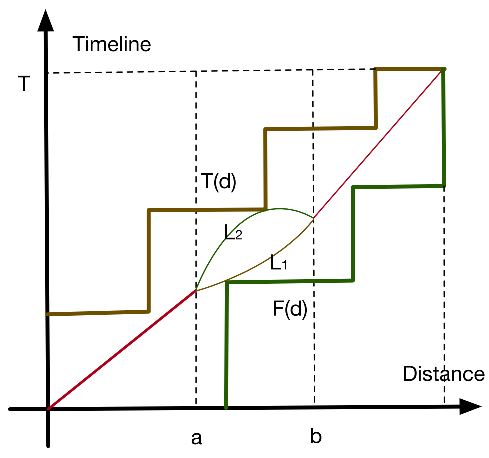


Fig. 12. Suppose L_2 and L_1 are two different solutions that obey Lemma 7 while satisfying formula (14) and (15) at the same time. However, the assumption is incorrect, for curve L_1 must be convex and curve L_2 has to be concave under the constraints of (14) and (15), which disobeys Lemma 3 and Lemma 5. Thus, solution satisfies Lemma 7 must be unique.

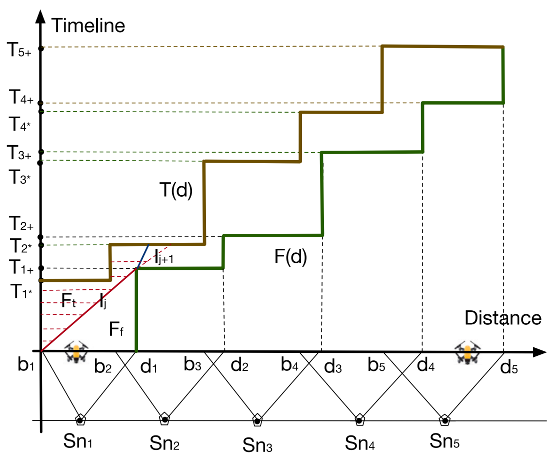


Fig. 13. We assume l_{j+1} is a solution of algorithm Offspeeding, which disobeys Lemma 3. For instance, the slope of line l_{j+1} satisfies $s_{j+1} > s_j$ when $T_j^+ = F(d_j)$. However, the assumption is incorrect, for l_{j+1} is included in set of previous solution l_j, where $l_{j+1} \in F_{tj}$ (marked in red horizon line), which indicates that l_{j+1} is not the solution of Offspeeding. Thus, solution of Offspeeding must obey Lemma 3.

general problem *EESS*, and it is proved to be optimal. On this basis, a heuristic online algorithm *EDS* is presented to solve the *EESS* problem in more realistic scenarios that information from IoT devices keeps unknown until the UAV flies close. Finally, simulation experiments are conducted to evaluate our proposed algorithms and the results demonstrate that our online algorithm *EDS* is near optimal.

In the future, a fine-grained energy consumption model is considered including communication and computing energy of the UAV. On this basis, we aim to minimize the energy consumption of the UAV under the deadline constraints by designing an optimal trajectory and proper flight speeds.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of Lemma 1

Proof. Since $p(v) = c_1 * v^3 + \frac{c_2}{v}$, we have $p(v)'' = (c_1 * v(d)^3 + \frac{c_2}{v(d)})'' = 6c_1 * v(d) + \frac{2*c_2}{v(d)^3} > 0$, so it is a convex function. ■

Appendix B. Proof of Lemma 3

Proof. We prove by contradiction. Suppose curve l_{opt}^L increases its slope that consumes less energy than decreases its slope. An example is given in Fig. 6, in which the accumulation curves l_1, l_2 and l_3 decrease their slopes. Assume the optimal curves l_{opt}^L is different from l_1, l_2, l_3, e.g., the UAV flies in curves l_1', l_2', l_2, l_3, where slope of l_2', defined as s_2' is larger than s_1', the slope of sub-line l_1'. Obviously, we have inequation $s_2' > s_1 > s_1'$. According to Lemma 2 and Corollary 1, the total energy consumption of l_1' and l_2' is larger than that of l_1, indicating that curve L_{opt}^L is not the optimal when the slope increases. ■

Appendix C. Proof of Lemma 4

Proof. We prove Lemma 4 by contradiction. Suppose curve l_{opt}^L does not intersect with lower bound $F(d)$ and consumes less energy. An example is given in Fig. 6, in which the accumulation curves l_1, l_2 and l_3 decrease their slopes. Assume the optimal curves l_{opt}^L different from l_1, l_2, l_3, e.g., the UAV flies in curves l_4' and l_5' that does not intersect with lower bound $F(d)$. Obviously, we can find another existing sub-line l_5' between l_3' and l_4', whose slope satisfies $s_3' > s_5' > s_4'$, According to Lemma 2 and Corollary 1, the total energy consumption of l_3' and l_4' is larger than that of l_5', indicating that curve L_{opt}^L is not the optimal when l_{opt}^L does not intersect with lower bound $F(d)$.

Appendix D. Proof of Lemma 8

Proof. We assume that there exists more than one solution that satisfies Lemma 7. Let l_1 and l_2 be two different curves in solution L, l_1 and l_2 are the same in $[0, a]$ and $[b, D]$, but different in (a, b), where:

$$\begin{cases} l_1(d) = l_2(d) & 0 \le d \le a, b \le d \le D \\ l_1(d) \ne l_2(d) & a < d < b \end{cases} \tag{14}$$

Without loss of generally, we assume $l_1(i) \le l_2(i)$, $i \in (a, b)$, thus

$$F(i) \le l_1(i) < l_2(i) \le T(i), \quad i \in (a, b) \tag{15}$$

For the assumption, $l_1(a) = l_2(a)$ and $l_1(i) > l_2(i)$ when $i \in (a, b)$, and $l_1(b) = l_2(b)$, shown in Fig. 12.

Based on Lemma 4, $l_1(i)$ must intersect with $F(i)$ when $i < b$, for $F(i) \le l_1(i) \le l_2(i) \le T(i), i \in (a, b)$. On the other hand, $l_2(i)$ must intersect with $T(i), i \in (a, b)$ based on Lemma 6. To let $l_1(b) = l_2(b)$, $l_1(i)$ must be a convex function and $l_2(i)$ must be a concave function, which violates Lemma 3 and Lemma 5.

Therefore, it is impossible for l_1 and l_2 to both obey Lemma 7 under constraints of (14) and (15) , which is contradicted to the assumption, which proves that solution obeys Lemma 7 must be unique. ■

Appendix E. Proof of Theorem 3

Proof. Lemma 8 proves that if there exists an algorithm that satisfies Lemmas 3–6, it must be an optimal algorithm for *EESS* problem.

Thus, this part proves Algorithm Offspeeding satisfies Lemmas 3–6.

For Lemmas 4 and 6, Algorithm Offspeeding results intersect either with $F(d)$ or with $T(d)$ (line 7 and line 12 in Offspeeding).

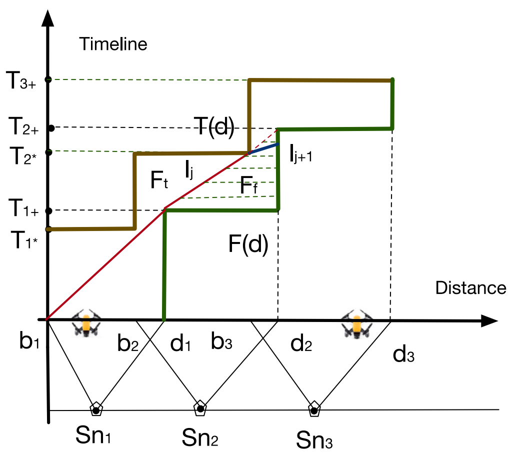


Fig. 14. We assume l_{j+1} is a solution of algorithm Offspeeding, which disobeys Lemma 5. For instance, the slope of line l_{j+1} satisfies $s_{j+1} < s_j$ when $T_j^+ = T(b_j)$. However, the assumption is incorrect, for l_{j+1} is included in set of previous solution l_j, where $l_{j+1} \in F_{fj}$ (marked in green horizon line), which indicates that l_{j+1} is not the solution of Offspeeding. Thus, solution of Offspeeding must obey Lemma 5.

For Lemma 3, we assume a counter-example that Offspeeding intersects $F(d)$ in iteration j, and the slope of next iteration satisfies $s_{j+1} > s_j$ (shown in Fig. 13) . In this scenario, the new solution L_{j+1} will be included in previous upper bound area F_{tj} (Marked as red line), for slope of L_{j+1} is s_{j+1} and satisfies $s_{j+1} > s_j$, which is incorrect. Thus, this scenario is impossible, and Algorithm Offspeeding obeys Lemma 3.

For Lemma 5, we also assume a counter-example that Offspeeding intersects with $T(d)$ in iteration j, and the slope of next iteration satisfies $s_{j+1} < s_j$ (shown in Fig. 14). In this scenario, the new solution L_{j+1} will be included in previous upper bound area F_{fj} (Marked as red line), for slope of L_{j+1} is s_{j+1} and satisfies $s_{j+1} < s_j$, which is incorrect. Thus, this scenario is impossible, and Offspeeding obeys Lemma 5.

Overall, Offspeeding obeys Lemmas 3–6, thus it must be the optimal algorithm.

In each iteration of algorithm Offspeeding, we attempt to find the longest feasible sub-line starting from the origin point, and the destination of this sub-line is the start point of the next iteration. Then, for the worst case, the UAV starts from the origin point and unfortunately tests all the other points before finding the suitable sub-line, i.e., the nearby point from the origin point. Under this circumstance, total number of iterations should be $(2m-1)+(2m-2)+\cdots+1$, which is $2m^2-m$. Therefore, the complexity of algorithm Offspeeding is $O(m^2)$. ■

Appendix F. Minimum-propulsion-power speed of UAV

Proof. Denote v^* as the UAV's minimum-propulsion-power flight speed. Based on Eq. (1), power consumption of UAV is:

$$p(v) = c_1 * v^3 + \frac{c_2}{v}$$

Since Appendix A has proved the equation to be convex, the derivation of equation is:

$$p'(v) = (3c_1 * v^2 - \frac{c_2}{v^2})$$

Let $p'(v) = 0$, then the optimal flight speed v^* is:

$$v^* = \sqrt[4]{\frac{c_2}{3c_1}} \quad \blacksquare$$

References

[1] Z. Yu, Y. Gong, S. Gong, Y. Guo, Joint task offloading and resource allocation in UAV-enabled mobile edge computing, IEEE Internet Things J. 7 (4) (2020) 3147–3159.

[2] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, Joint computation and communication design for UAV-assisted mobile edge computing in IoT, IEEE Trans. Ind. Inf. 16 (8) (2020) 5505–5516.

[3] H. Guo, J. Liu, UAV-enhanced intelligent offloading for internet of things at the edge, IEEE Trans. Ind. Inf. 16 (4) (2020) 2737–2746.

[4] J. Zhang, L. Zhou, Q. Tang, E.C. Ngai, X. Hu, H. Zhao, J. Wei, Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing, IEEE Internet Things J. 6 (2) (2019) 3688–3699.

[5] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, X. Shen, Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization, IEEE Trans. Veh. Technol. 69 (3) (2020) 3424–3438.

[6] S. Jeong, O. Simeone, J. Kang, Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning, IEEE Trans. Veh. Technol. 67 (3) (2018) 2049–2063.

[7] Qiyu Hu, Yunlong Cai, Guanding Yu, Zhijin Qin, Minjian Zhao, Geoffrey Ye Li, Joint offloading and trajectory design for UAV-enabled mobile edge computing systems, IEEE Internet Things J. 16 (99) (2018) 476–490.

[8] Xiaowen Cao, Jie Xu, Rui Zhang, Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization, in: 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2018, pp. 1–5.

[9] X. Hu, K. Wong, K. Yang, Z. Zheng, UAV-assisted relaying and edge computing: Scheduling and trajectory optimization, IEEE Trans. Wireless Commun. 18 (10) (2019) 4738–4752.

[10] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, M. Pan, IoT enabled UAV: Network architecture and routing algorithm, IEEE Internet Things J. 6 (2) (2019) 3727–3742.

[11] X. Diao, J. Zheng, Y. Cai, Y. Wu, A. Anpalagan, Fair data allocation and trajectory optimization for UAV-assisted mobile edge computing, IEEE Commun. Lett. 23 (12) (2019) 2357–2361.

[12] Dimitrios Sikeridis, Eirini Eleni Tsiropoulou, Michael Devetsikiotis, Symeon Papavassiliou, Wireless powered public safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency, J. Netw. Comput. Appl. 123 (2018) 69–79.

[13] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, Y. Liu, Multi-UAV-enabled load-balance mobile-edge computing for IoT networks, IEEE Internet Things J. 7 (8) (2020) 6898–6908.

[14] R. Duan, J. Wang, J. Du, C. Jiang, T. Bai, Y. Ren, Power-delay trade-off for heterogenous cloud enabled multi-UAV systems, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–6.

[15] Yuwen Qian, Feifei Wang, Jun Li, Long Shi, Feng Shu, User association and path planning for UAV-aided mobile edge computing with energy restriction, IEEE Wirel. Commun. Lett. WCL (2019).

[16] Xiaoyan Hu, Kai Kit Wong, Kun Yang, Zhongbin Zheng, UAV-assisted relaying and edge computing: Scheduling and trajectory optimization, IEEE Trans. Wireless Commun. PP (99) (2019) 1.

[17] Y. Liu, K. Xiong, Q. Ni, P. Fan, K.B. Letaief, UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, cpu control, and trajectory optimization, IEEE Internet Things J. 7 (4) (2020) 2777–2790.

[18] Yong Zeng, Qingqing Wu, Rui Zhang, Accessing from the sky: A tutorial on UAV communications for 5G and beyond, Proc. IEEE 107 (12) (2019) 2327–2375.

[19] Zeng Yong, Zhang Rui, Energy-efficient UAV communication with trajectory optimization, IEEE Trans. Wireless Commun. 16 (6) (2017) 3747–3760.

[20] C. Zhang, W. Zhang, Spectrum sharing for drone networks, IEEE J. Sel. Areas Commun. 35 (1) (2017) 136–144.

[21] S. Eom, H. Lee, J. Park, I. Lee, UAV-aided wireless communication design with propulsion energy constraint, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[22] J. Xu, Y. Zeng, R. Zhang, UAV-enabled wireless power transfer: Trajectory design and energy optimization, IEEE Trans. Wireless Commun. 17 (8) (2018) 5092–5106.

[23] Y. Liu, K. Xiong, Q. Ni, P. Fan, K.B. Letaief, UAV-assisted wireless powered cooperative mobile edge computing: joint offloading, cpu control, and trajectory optimization, IEEE Internet Things J. 7 (4) (2020) 2777–2790.

[24] F. Zhou, Y. Wu, R.Q. Hu, Y. Qian, Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems, IEEE J. Sel. Areas Commun. 36 (9) (2018) 1927–1941.

[25] Yong Wang, Zhi-Yang Ru, Kezhi Wang, Peiqiu Huang, Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing, IEEE Trans. Cybern. PP (2019) 1–14.

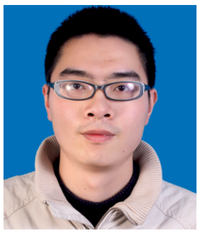
[26] C.D. Franco, G. Buttazzo, Energy-aware coverage path planning of uavs, in: 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, 2015, pp. 111–117.

[27] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Merouane Debbah, Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications, IEEE Trans. Wireless Commun. 16 (11) (2017) 7574–7589.

[28] Y. Zeng, J. Xu, R. Zhang, Energy minimization for wireless communication with rotary-wing UAV, IEEE Trans. Wireless Commun. 18 (4) (2019) 2329–2345.

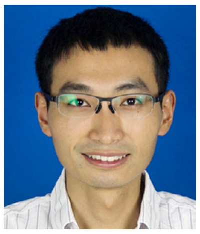
[29] Mohamed Alzenad, Amr El-Keyi, Faraj Lagum, Halim Yanikomeroglu, .3D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage, IEEE Wirel. Commun. Lett 6 (4) (2017) 434–437.

[30] A. Trotta, F.D. Andreagiovanni, M. Di Felice, E. Natalizio, K.R. Chowdhury, When UAVs ride a bus: Towards energy-efficient city-scale video surveillance, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1043–1051.
[31] Murtaza A. Zafer, Eytan Modiano, A calculus approach to energy-efficient data transmission with quality-of-service constraints, IEEE/ACM Trans. Netw. 17 (3) (2009) 898–911.
[32] W. Wu, J. Wang, M. Li, K. Liu, F. Shan, J. Luo, Energy-efficient transmission with data sharing in participatory sensing systems, IEEE J. Sel. Areas Commun. 34 (12) (2016) 4048–4062.


Weidu Ye received the B.S. degree in Computer Science from Nanjing Forestry University in 2015. He is currently working towards the Ph.D. degree in the School of Computer Science and Engineering in Southeast University, Nanjing, China. His research interests include edge computing and UAV-assisted networking.

Junzhou Luo received the B.S. degree in applied mathematics and the MS and Ph.D. degrees in computer network, all from Southeast University, China, in 1982, 1992, and 2000, respectively. He is a full professor in the School of Computer Science and Engineering, Southeast University. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and he is a member of the ACM and chair of ACM SIGCOMM China. His research interests are next generation network architecture, network security, cloud computing, and wireless LAN.

Feng Shan received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2015. He is currently an associate professor with the School of Computer Science and Engineering, Southeast University. He was a Visiting Scholar with the School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO, USA, from 2010 to 2012. His current research interests include energy harvesting, wireless power transfer, UAV-assisted networking, swarm intelligence, and algorithm design and analysis.


Wenjia Wu received the B.S. and Ph.D. degrees in computer science in 2006 and 2013, respectively, from Southeast University. He is an associate professor at the School of Computer Science and Engineering in Southeast University. His research interests include wireless and mobile networks.

Ming Yang received the Ph.D. degree in computer science from Southeast University, Nanjing, in 2007. Currently, he is an associate professor at the School of Computer Science and Engineering in Southeast University, Nanjing, China. His research interests include network security and privacy. He is a member of CCF and ACM, as well as deputy director of Key Laboratory of Computer Network and Information Integration, Ministry of Education.