Energy-Efficient General PoI-Visiting by UAV with a Practical Flight Energy Model

Feng Shan, *Member, IEEE,* Jianping Huang, Runqun Xiong, *Member, IEEE,* Fang Dong, *Member, IEEE,* Junzhou Luo, *Member, IEEE,* and Suyang Wang, *Member, IEEE,*

Abstract—Unmanned aerial vehicles (UAVs) are being widely exploited for various applications, *e.g.,* traversing to collect data from ground sensors, patrolling to monitor key facilities, moving to aid mobile edge computing. We summarize these UAV applications and formulate a problem, namely the *general waypoint-based PoI-visiting problem*. Since energy is critical due to the limited onboard storage capacity, we aim at minimizing flight energy consumption. In our problem, we pay special attention to the energy consumption for turning and switching operations on flight planning, which are usually ignored in the literature but play an important role in practical UAV flights according to our real-world measurement experiments. We propose specially designed graph parts to model the turning and switching cost and thus transfer the problem into a classic graph problem, *i.e.,* general traveling salesman problem, which can be efficiently solved. Theoretical analysis shows that such problem transformation has the graph redefinition approximation ratio upper bound, $max\{\Theta/\delta, 2\}$, where Θ is related to the designed graph parts and δ is a constant. Finally, we evaluate our proposed algorithm by simulations. The results show that it costs less than 107% of the optimal minimum energy consumption for small scale problems and costs only 50% as much energy as a naive algorithm for large scale problems.

Index Terms—Unmanned Aerial Vehicle, Energy Efficient, Path Planning, Graph Theory

1 Introduction

Unmanned aerial vehicles (UAVs), especially rotor-wing UAVs, are becoming increasingly popular because they are more and more affordable. They are being exploited widespreadly for various applications, *e.g.,* traversing to collect data from ground sensors [1]–[8], patrolling to monitor key ground facilities [9]–[13], moving to aid ground mobile edge computing [14]–[18]. Compared with traditional ground robots [19], which have to avoid countless obstacles or otherwise restricted to given routes (road or rail), UAVs are more agile and flexible in mobility.

UAV flight planning plays an important role in these UAV-aided applications. We discover that automatic flight planning is mostly implemented by waypoints, for both community-supported open-source UAVs [20] and commercial closed-source UAVs [21]. This is because waypoint-based path planning is quite simple yet robust, hence supported by almost every programmable UAV. In other words, modern rotor-wing UAVs plan their route by a serial of waypoints, which is a location where UAVs stop and make turns. Therefore, waypoints divide the flight route into a serial of straight line path segments, where a UAV accelerates, maintains the cruise speed, and decelerates.

By investigating the UAV flight planning for popular UAV applications, such as data collection, surveillance and monitor, UAV-aided edge computing, we formulate a general waypoint-based Points of Interests (PoIs) visiting application scenario that a UAV is planned to fly a waypoint-based flight route and visit a set of PoIs. In our general scenario, a UAV visits a PoI by visiting any point within its range, and even the ranges of PoIs may overlap, so it is possible to visit more than one PoIs from the same waypoint, with additional switching energy cost introduced later.

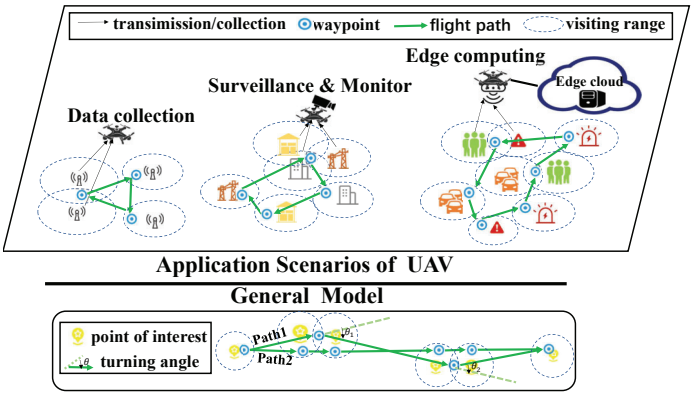


Fig. 1. A general PoI-visiting problem is formulated from various application scenarios of UAV, such as data collection, surveillance and monitor, UAV-aided edge computing.

As shown in Fig. 1, such a generalized application scenario matches various applications. (1) In the data collection application, a UAV traverses all deployed ground sensors to collect data and each ground sensor has a transmission range, within which the UAV can receive the sensed data. The transmission ranges may

- *F. Shan, R. Xiong, F. Dong, and J. Luo are with School of Computer Science and Engineering, Southeast University, Nanjing 210096, Jiangsu, P. R. China (Emails: {shanfeng, rxiong, fdong, jluo}@seu.edu.cn)*
- *J. Huang is with Huawei Technologies Co. Ltd, Nanjing 210012, Jiangsu, P. R. China. (Email:huangjianping12@huawei.com).*
- *S. Wang is with School of Computer Science and Engineering, Southeast University, Nanjing 210096, Jiangsu, P. R. China, and aslo with Jiangsu Jinheng Information Technology Co., Ltd, Nanjing 210035, Jiangsu, P. R. China. (Email:wangsy@njsteel.com.cn).*

Manuscript received xx Xxx. 202x; revised xx Xxx. 202x; accepted xx Xxx. 202x. Date of publication xx Xxx. 202x; date of current version xx Xxx. 202x. (Corresponding authors: Feng Shan and Fang Dong.)
Digital Object Identifier no. 10.xxxx/TMC.202x.xxxxxxx

overlap with each other, while in the common range area, the UAV has to switch from receiving one sensor to receiving another. (2) In the surveillance and monitor application, a UAV patrols a set of key facilities with an optical equipment and each facility has a visibility range, only within which the onboard optical camera can sense useful data. The visibility ranges may overlap when two facilities are close, but the UAV must switch by rotating the camera from one facility to another. (3) In the UAV-aided edge computing application, a UAV equipped with a powerful computation unit is dispatched to aid ground device computation and each ground device has a computation offloading range, within which the UAV can receive the offloaded computation tasks. Such offloading ranges may overlap with each other, while the UAV has to receive offloading tasks sequentially.

Although many work [1], [9], [22], [23] have studied some of these popular UAV applications, most of them investigate the UAV flight planning for special application scenarios only, *e.g.,* the algorithm proposed in [1] is for collecting data, the method designed in [9] is for monitoring in severe environment, and the solution provided in [22] is intended for automating CSI map construction, which is difficult to extend these algorithms to other application scenarios. Moreover, some existing researches view the ground PoI as a single "point" rather than a range, failing to model general problems in real-world scenarios. We thus formulate a UAV flight planning that generalizes a number of UAV applications.

Besides, energy is an important consideration for UAV flight planning. Due to limitation of the storage capacity onboard, the UAV energy consumption directly affects its flight endurance. Previous work has concluded [24] that, for a typical commercial UAV, the flight energy consumption accounts for the most proportion than any other operations, such as wireless transmission. However, the previous studies on UAV flight energy models [1], [7], [25], [26] simplify UAV flight operations and do not reflect energy consumption accurately in models, which motivates us to propose an improved and practical flight energy model.

To reveal a more practical and refined energy consumption model for waypoint-based UAV flight planning, we conduct a set of real-world experiments than the previous researches [27]–[29]. By our experiment results, we disclosed that, in addition to the distance covered, making turns and switching PoIs-visiting also affect energy consumption. More specifically, the larger turn a UAV makes at a waypoint, the more flight energy consumed; the more times it switches at a waypoint (to visit more PoIs), the more energy consumption. Details will be introduced in Section 3. Our energy model considering the energy consumption for turning and switching is hence distinct from the most existing researches.

The problem to find an energy-efficient flight path planning that considers the additional turning and switch energy is called the *general waypoint-based PoI-visiting* problem. This problem is quite challenging. The readers can sense the challenges from two example paths illustrated in Fig. 1. We can see that both Path 1 and Path 2 are composed of a serial of waypoints and straight line path segments. A UAV stops and makes a turn at a waypoint, so during the straight line path flight, a UAV accelerates, maintains the cruise speed, and decelerates. As a result, Path 2 consumes more energy on acceleration and deceleration, because it has more waypoints than Path 1. However, a UAV following Path 1 makes larger angle turns and has to switch more between PoIs at a waypoint, hence, it spends more energy on turning and switching. Moreover, Path 1 is longer, so energy consumption in covering distance is more than that of Path 2.

In summary, for any UAV flight planning algorithm to solve our problem, a two-fold challenging tradeoff is unignorable. On the one hand, there is a challenging tradeoff among the number of waypoints, the size of turning angle, and the times of switching. Fewer waypoints usually means that most waypoints of the route are in overlapping range which leads to more switching between PoIs and more winding flight path with larger angle of turns. On the other hand, there is a challenging tradeoff among the number of waypoints, the acceleration/deceleration, and the flight distance. Fewer waypoints usually cause fewer straight line path segment in the route, and thus fewer acceleration and deceleration; while fewer waypoints also mean more overlapping-waypoints, and the UAV usually has to detour to visit these waypoints with longer flight distance due to the small overlapping-range.

As the challenges stated above, it is hard to find a straightforward solution to the *general waypoint-based PoI-visiting* problem. Although much research effect has already been put to solve similar UAV path planning problems, none of these existing methods can be applied directly. First, theoretical methods based on conventional mathematical programming can not be applied, because our restrictions on turning angle and switching times are non-traditional, so a solution, if exists, may have scalability issues. Second, machine learning techniques do not work either, because the solution efficiency relies on empirical and training models which is dedicated to a specific problem, while our problem are intended to be general. As a result, we tackle this problem from a direct angle. Since, the partial goal of the *general waypoint-based PoI-visiting* problem is to minimize the covered distance of a tour, which can be naturally associated with the classic traveling salesman problem (TSP) and its variants that aim to find a minimum-cost cycle on a graph, so this motivates us to propose a graph based solution to this problem. The straight flight cost can be easily reflected in a graph, where the turning cost (proportional to the turn angle) and the switching cost (proportional to the times of switching) can not be embedded, which is the main gap we need to bridge in our proposed graph based solution.

Therefore, the contributions of this paper are summarized as follows.

- We generalize a number of popular UAV applications to formulate a general UAV flight planning problem, named the *general waypoint-based PoI-visiting* problem, aiming at minimizing the UAV flight energy consumption.
- We devise a set of real-world experiments, and our new findings include that turning at a waypoint and switching between PoIs also cost energy, based on which, we develop a more practical and refined flight energy model for rotor-wing UAVs.
- We propose a novel graph based approach that the turning and switching cost are respectively modelled by tactful regular polygons and virtual splitting, to convert the original problem into a classic graph problem which can be solved efficiently. Theoretical analysis shows that the graph redefinition approximation ratio is $max\{\Theta/\delta, 2\}$, where Θ is related to the designed graph parts and δ is a constant.
- We conduct simulations to evaluate the performance of the proposed algorithm. The results show it performs near the optimal solution, within 107% of the minimum energy consumption for small scale problems, and costs only 50%

the energy by a naive algorithm for large scale problems.

The rest of the paper is organized as follows. Section 2 surveys related work. Our motivation is presented in Section 3. Section 4 shows the system model and problem formulation. And a novel approach is proposed in Section 5. Then the theoretical analysis and solution to the problem are given in Section 6. The elaborate simulations are introduced in Section 7. Finally, Section 8 concludes the paper.

2 Related Work

2.1 Theoretical methods for UAV flight path planning

The theoretical methods for UAV flight path planning are the basis of applications to guarantee the efficiency and safety for practice. There are extensive researches that provide valuable ideas in solving path planning problems. **(a)** Much work utilizes conventional mathematical programming to solve path planning problems [30]–[33]. Zeng *et al.* [30], [31] and Zhan *et al.* [32] use successive convex optimization to optimize the UAV flight path. To solve a nonlinear optimization UAV path planning problem, Shi *et al.* [34] transform this problem into an efficiently solvable integer linear programming subproblem by relaxing some constraints. Based on gradient-based trajectory optimization, Zhou *et al.* [33] devise a path-guided optimization to tackle infeasible local minima, which improves the path replanning success rate significantly. Iteration based numerical solvers for such mathematical programming is time consuming if the desired accuracy is high; while other heuristic solvers for programming that involves mutual restrictions or non-convex cases may have scalability issues. **(b)** The learning-based algorithms are recently popular in UAV flight path planning optimization [35]–[38]. For instance, in [35], [36], the authors leverage deep learning approach to design the UAV flight path in data collection. The authors in [37] propose an effective UAV path planning approach based on reinforcement learning for satisfaction-aware data offloading in surveillance systems. Then [38] also refers to reinforcement learning technique to determine the optimal UAV trajectories. However, machine-learning based solutions' efficiency relies on empirical and training models which are dedicated to a specific problem, while our problem are intended to be general. **(c)** Therefore, the graph-based method attracts our attention. Like in [22], [39], the authors transform the original path planning problems into traveling salesman problem by constructing a graph to obtain the well-performing flight path planning. Nevertheless, these existing algorithms are only applicable to specific scenarios, we thus propose a graph-based method to the general flight path planning in this work.

2.2 Energy-efficient UAV-aided applications

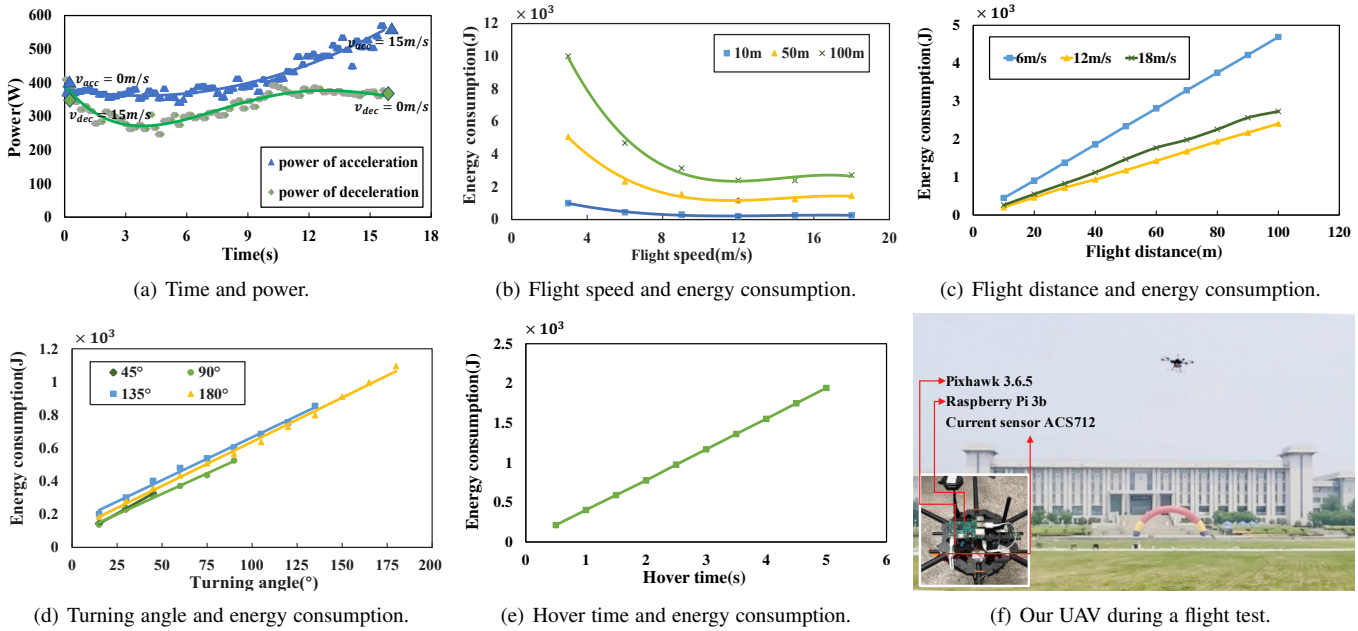
In this subsection, we give a brief introduction of energy-efficient UAV-aided application researches, *e.g.,* (a) UAV-aided ubiquitous coverage to provide seamless wireless coverage service [40], [41], (b) UAV-aided information/data collection to make up for the shortcomings of the traditional network [5], [42], [43] and (c) UAV-aided edge computing or relaying to improve the stability and connectivity of the system [15], [44], [45]. **(a)** When UAVs are exploited to enhance communication coverage, Liu *et al.* [40] and Zhang *et al.* [41] emerge the deep reinforcement learning to devise an energy-efficient UAV trajectory. **(b)** For data collection, Ghorbel *et al.* [42] use classic linear programming and heuristic algorithm to design efficient-energy path planning of data collection. To maximize the accumulative volume of collected data, Chen *et al.* [5] design a $(1-1/e)$-approximation algorithm for the energy-constrained UAV. Focused on the security and efficiency of data collection, Xu *et al.* [43] integrate an adaptive linear prediction algorithm into blockchain-enabled system for data collection scenario. **(c)** Other applications are in edge computing scenario, where the time constraints are much stringent. Tun *et al.* [15] and Li *et al.* [44] propose energy-efficient methods based on mathematical optimization *e.g.,* block successive upper-bound minimization and successive convex optimization, to plan UAV path of edge computing with computation latency, transmit power and communication requirements. And the popular deep reinforcement learning also used in a considerable number of applications. For instance, Wei *et al.* [45] propose a distributed deep reinforcement learning based method with the cooperative exploring and prioritized experience replay to solve the practical UAV-assisted computation offloading problem under changing environment. As many UAV-aided applications are emerging, a more general problem that fits multiple application scenarios is desired.

2.3 UAV flight energy model

In this subsection, we focus on the UAV flight energy model, which is the basis for the design of energy-efficient flight path planning. In fact, UAV flight energy models can be loosely classified into three categories: (a) the distance-related model [24], [25], [40], [46], (b) the duration-related model [1], [4], and (c) the speed-related model [15], [47]–[49]. **(a)** Some researchers prefer the distance-related model. Ahmed *et al.* [24] obtain the distance-related model for a fleet of UAVs. Subsequently, Liu *et al.* [40] and Xiong *et al.* [46] in UAV energy consumption optimization problems use the distance-related model. And for the control and monitoring platform of heterogeneous UAVs, Huang *et al.* [25] still adopt the distance-related flight energy model. **(b)** Considering that the effect of distance on energy consumption is not well measured due to the variability of UAV, hence there are also extensive efforts that adopt the duration-related model, but only focus on the duration of UAV. For example, Mozaffari *et al.* [4] and Gong *et al.* [1] choose the UAV duration-related energy model in data collection scenario. In fact, although duration-related energy model has its justification in the energy optimization problem, it is still too simple to present the complex mobility of UAVs. **(c)** Some researches indicate that the speed-related model is more practical. For example, Morbidi *et al.* [47] obtain a speed-related model and determine the minimum-energy paths for UAV. More recently, Tun *et al.* [15] utilize speed-related model to solve UAV path planning problem. However, these speed-related models only partly characterize the UAV flight energy, and lack practicality. This motivates us to remodel UAV energy model through real-world experiments extended from our previous work [50]. Our resulting energy model follows a theoretical speed-related energy model developed by Zeng *et al.* [48], [49]. We will describe in detail the experiments on the flight energy model in the next section.

3 Motivation

To obtain a more practical and accurate UAV energy consumption model for waypoint-based flight planning, we conduct a series of real-world experiments. In this section, we report the detailed



(a) Time and power. (b) Flight speed and energy consumption. (c) Flight distance and energy consumption.

(d) Turning angle and energy consumption. (e) Hover time and energy consumption. (f) Our UAV during a flight test.

Fig. 2. A practical energy consumption model based on our real-world flight tests. (a) presents the trends of power in different times during acceleration/deceleration. In (b), there is an optimal speed to minimize the energy consumption for a fixed distance. In (c), it is apparent that the longer distance, the more energy consumed. (d) illustrates that energy consumption is a linear function about the turning angle. (e) shows that the longer time UAV hovers, the more energy consumed. (f) is a photo of our UAV during a field flight test.

experiment settings and results. The experiment results show that varying flight speed and making turns also affect energy cost, which motivates the study of this paper.

In the field flight tests, we use a hexacopter rotor-wing UAV, Model X4108 with 3.8 kg and a 1000mAh-capacity battery. This UAV is equipped with autopilot Pixhawk 3.6.5 flight controller, and a companion computing device, Raspberry Pi 3b single-board computer (RPi). The real-time voltage value is read by the RPi through USB from the flight controller via the MAVLink communication protocol. A current module ACS712 is installed onboard to detect the real-time battery current, which can be read by the RPi via the I^2C communication protocol. Given the real-time current and voltage, it is easy to calculate the real-time power consumption. Furthermore, the companion RPi sends control commands by MAVLink protocol to UAV flight controller automatically via the USB link.

In our field flight tests, we focus on the energy consumption 1) between two waypoints and 2) at a single waypoint. More specifically, for the straight line flight path between two waypoints, we test energy consumption for 1.1) acceleration/deceleration, 1.2) distance and 1.3) cruise speed flight; for the operation at a single waypoint, we test energy consumption for 2.1) making turns and 2.2) hovering to switch PoI-visiting. Field test results are given in Fig. 2(a)-(e), and Fig. 2(f) shows our UAV during a flight test.

Field test 1.1: the relationship between time and flight energy power during acceleration/deceleration. In this test, we let the UAV accelerate (decelerate) at $1\mathrm{m/s}^2$ for 16 seconds, from 0 $\mathrm{m/s}$ (16 $\mathrm{m/s}$) at the 0th second to 16 $\mathrm{m/s}$ (0 $\mathrm{m/s}$) at the 16th second. The time-power curves are shown in Fig. 2(a). Based on the analysis of the result data, we obtain the function of the motor power P in acceleration is related to flight time t through $P = 1.3876t^2 - 10.591t + 381.79$. Similarly, we obtain the relation between P and t in deceleration as a function $P = 0.0114t^4 - 0.7279t^3 + 12.845t^2 - 69.958t + 389.74$. Apparently, the acceleration/deceleration consumes energy, so more waypoints in UAV flight planning path means more acceleration/deceleration, which results to inevitable energy consumption.

Field test 1.2: the relationship between flight speed and flight energy power. In this test, we let the UAV fly straight at various cruise speeds for a distance of 10, 50 and 100 meters respectively. We test 6 cruise speeds for each kind of distance, and the relationship between UAV energy consumption E and flight speed v is shown in Fig. 2(b). We learn that given a fixed distance there exists an optimal speed (not necessarily the maximum speed) for the minimum energy consumption, and this finding indeed satisfies a recent theoretical study [49].

Field test 1.3: the relationship between flight distance and flight energy consumption. In this test, we control the UAV to fly a fixed distance of 100 meters, at cruise speed v of 6 $\mathrm{m/s}$, 12 $\mathrm{m/s}$ and 18 $\mathrm{m/s}$ respectively, and measure the real-time energy consumption E. From the results in Fig. 2(c), we have $E = 47.249d$ when $v = 6$ $\mathrm{m/s}$, $E = 27.836d$ when $v = 18$ $\mathrm{m/s}$, and $E = 24.456d$ when $v = 12$ $\mathrm{m/s}$. There is an optimal speed, 12 $\mathrm{m/s}$, which verifies the conclusion in Field test 1.2. And the total energy cost to fly $100m$ at 12 $\mathrm{m/s}$ is $2445J$, so the distance-covering flight energy cost is an important part of the flight energy consumption.

Field test 2.1: the relationship between turning angles and flight energy consumption. In this test, we command the UAV to make a turn at an angle of 45°, 90°, 135°, and 180° respectively. The energy consumption is illustrated in Fig. 2(d). It is easy to see that the angle θ and the UAV energy consumption E are related nearly through function $E = 5.3316\theta + 104.65$ in our settings. When the UAV makes a 45° turn, its energy cost reaches close to $300J$; when the UAV makes a 180° turn, its energy cost reaches close to $1100J$, as a conclusion, the turning cost is non-negligible, and related to the turning angle rather than replaced by a fixed cost as [22].

Field test 2.2: the relationship between hover time and flight energy consumption. In this test, we vary the hovering time and obtain the UAV energy consumption. The results are shown in Fig. 2(e). The UAV energy consumption E is linear with the hovering time t. In our settings, we have $E = 389.15t$. So, as the hovering time increases to $5s$, the UAV has to consume almost $2000J$ energy. Therefore, in our general energy model, if the UAV hovers for a while to adjust itself for better visiting, this energy cost can not be ignored.

These findings from the field test results motivate us to redefine UAV flight energy consumption composition, which will be described in detail in the next section. Although our energy model is motivated from the field tests of a specific UAV, the flight energy model we are about to build is general, because our model is designed for waypoint based rotor-wing UAVs.

4 System Model and Problem Formulation

This section formulates the *general waypoint-based PoI-visiting* problem, which generalizes some popular UAV application scenarios, such as data collection, monitoring and surveillance, and UAV-aided edge computing. We highlight our field-test-originated, more practical and more accurate UAV flight energy consumption model for waypoint-based flight planning.

4.1 System model

Assume there are n PoIs, denoted as PoI $i, i = 1, 2, \cdots, n$, where a PoI can be a wireless sensor, a facility, or a mobile device. These PoIs are randomly located within a rectangle region, and each PoI has a range, *e.g.,* the transmission range of a wireless sensor, the visibility range of a key facility, the computation offloading range of a mobile device. The area covered by the range of PoI i is denoted as R_i, which is usually a circle. Note that the radius of R_i is allowed to vary for different PoIs, and R_i may overlap with another range area R_j.

There is a base station within the region, and a UAV is dispatched from this station, flying along a planned tour to visit all PoIs, and returns to the original station eventually. Without loss of generality, we denote the base station as PoI 0. Note that PoI i is visited by the UAV if any point within area R_i is visited as a waypoint. In each tour, the UAV flies at a fixed altitude, and follows a waypoint-based route. Assume there are m waypoints on the route, denoted as Waypoint $j, j = 1, 2, \cdots, m$. Obviously, at Waypoint x, the UAV can visit PoI i only if R_i contains Waypoint x. Let D_x represent the set of PoIs that the UAV visits at Waypoint x. If $|D_x| > 1$, it means the UAV visits (motionlessly) more than one PoI at Waypoint x. Each PoI must be visited, so we have *all PoIs visiting constraint*,

$$|\bigcup_x D_x| = n + 1. \tag{1}$$

Since visiting a certain PoI can be equivalent to visiting any waypoint inside its range, we divide the region by small grid to reduce the searching space of the planning algorithm as shown in Fig. 3. Assume an $N \times M$ size region is discretized into multiple $g \times g$ size grids and a grid is represented by its grid center, called a *waypoint candidate*. Area R_i thus may cover a set of waypoint candidates, and let the set be denoted as S_i, which is called the *waypoint candidate set* for PoI i. Meanwhile, let S represents the collection of all waypoint candidate subsets, *i.e.,* $S = \{S_1 \cup S_2 \cup \cdots \cup S_n\}$. Only at a Waypoint $x \in S_i$, a UAV can visit PoI i, in other words, $i \in D_x$ implies that $x \in S_i$, so we have the *range visiting constraint*,

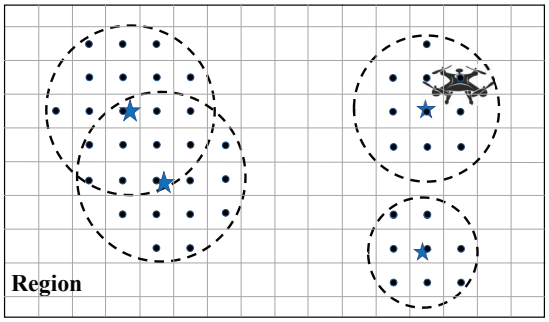


Fig. 3. The relationship among ranges, grids and waypoint candidates within the region. A set of square grids is used to discretize the region. PoIs are marked by blue stars, whose ranges are colored in black dashed circles, and the waypoint candidates are marked by black dots.

$$\forall i \in D_x \Rightarrow x \in S_i \quad \forall x. \tag{2}$$

A waypoint candidate $x \in S_i$ can be also covered by other PoIs $x \in S_j$ due to the overlapping, so we denote all these PoIs as set P_x,

$$P_x = \{i | x \in S_i\}. \tag{3}$$

An illustration of the relationship of ranges, grids, and waypoints is in Fig. 3. We want to plan a route that consists of a serial of waypoints to ensure all PoIs are visited, so planning a route is reduced to choosing waypoints from all waypoint candidates. Assume x and y are two arbitrary waypoint candidates of the region. Let w_{xy} indicts whether straight directed path e_{xy}, from x to y, is included in the route,

$$w_{xy} = \begin{cases} 1, & e_{xy} \text{ is in the route,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Then, when $|D_x| > 0$, there must be a path to and from x; when $|D_x| = 0$, certainly there is no such path. We have the following *route connecting constraint*,

$$\begin{cases} \sum_{\forall y} w_{yx} = \sum_{\forall y} w_{xy} \geq 1, & |D_x| > 0, \\ \sum_{\forall y} w_{yx} = \sum_{\forall y} w_{xy} = 0, & |D_x| = 0. \end{cases} \tag{5}$$

Our goal is to plan a route such that the UAV energy consumption is minimized. We list the key notations in Table 1.

TABLE 1
Definition of notation

Notation	Definition
R_i	The area covered by the range of PoI i
S_i	Waypoint candidate set of PoI i
D_x	Set of PoIs that the UAV visits at Waypoint x
P_x	Set of PoIs that covers waypoint candidate x
e_{xy}	Direct path from Waypoint x to Waypoint y
w_{xy}	If e_{xy} in the route then $w_{xy} = 1$, otherwise $w_{xy} = 0$
q_{xyz}	Angle between e_{xy} and e_{yz}
d_{xy}	Index of the flight direction from x to y, $d_{xy} = 1, 2, \cdots, 8$
$E(\cdot)$	Energy consumption
E_C	Overall energy consumption for straight flight
E_T	Overall energy consumption for making turns
E_S	Overall energy consumption for switching PoI-visiting
E_M	Overall energy consumption for service provisioning

4.2 Energy consumption model

We now model the energy consumption according to our real-world experiments in the previous section. Although our field tests are based on a specific UAV, the flight energy consumption model in this subsection is designed for waypoint based rotor-wing UAVs, which is general.

Let x and y be two arbitrary waypoint candidates. Flying straight from x to y, the UAV first accelerates to the cruise speed and then flies at this speed until it decelerates to 0 to visit Waypoint y.

Definition 1 (Straight flight energy consumption)**.** *The energy consumption for a straight flight on path e_{xy}, from waypoint candidate x to y, is defined as $E(e_{xy})$, which is related to acceleration/deceleration and cruise speed.*

Our proposed method can handle arbitrary energy function $E(e_{xy})$. In our simulations, we set $E(e_{xy}) = c_1|e_{xy}|+C_1$, where c_1 is the energy consumption ratio proportional to the distance, and C_1 is the energy consumption related to acceleration/deceleration. We define the total energy consumption related to straight flight by E_C,

$$E_C = \sum_{\forall x,\forall y} E(e_{xy})w_{xy}. \tag{6}$$

Let x, y and z be three arbitrary waypoint candidates that are not on a straight line. Between the two adjacent straight paths, e_{xy} and e_{yz}, the UAV has to make a turn at y. Denote the angle of turn as q_{xyz}, which is determined by the locations of x, y and z.

Definition 2 (Turning energy consumption)**.** *The energy consumption for a UAV to make a turn at waypoint candidate y, from x to z, is defined to be $E(q_{xyz})$, where q_{xyz} is the heading angle changed from path e_{xy} to e_{yz}.*

Energy function $E(q_{xyz})$ is the energy consumption related to the turning angle q_{xyz}. In our simulations, we set $E(q_{xyz}) = c_2q_{xyz} + C_2$, where c_2 and C_2 are constant factors related to a special UAV. The total energy consumption for all turns is denoted as E_T, which can be calculated as

$$E_T = \sum_{\forall x,\forall y,\forall z} E(q_{xyz})w_{xy}w_{yz}. \tag{7}$$

If the UAV switches PoI-visiting at Waypoint x, *i.e.,* D_x contains more than one PoI, additional energy consumption occurs because the UAV is required to hover to set out the switching, *e.g.,* establishing connection to a mobile device or sensor, rotating the optical camera from one direction to another, and the cost is called switching energy consumption.

Definition 3 (Switching energy consumption)**.** *The energy consumption for a UAV to switch between PoIs at waypoint candidate x is defined to be $E(D_x)$, related to the number of PoIs in D_x.*

If $|D_x| = 1$, there is no switch needed, we set the switching energy to be proportional to $|D_x|-1$, *i.e.,* $E(D_x) = c_3(|D_x|-1)$, where c_3 is the constant factor that related to a special UAV. So the total energy consumption of switching between PoIs is aggregated by all waypoints, denoted as E_S, which can be calculated as

$$E_S = \sum_{\forall x} E(D_x). \tag{8}$$

Moreover, the UAV generally needs additional energy consumption to provide service for PoIs, *i.e.,* data collection, recording video in monitoring and computation in edge computing, and this cost is called service energy consumption.

Definition 4 (Service energy consumption)**.** *The energy consumption for a UAV to provide service for a PoI i is denoted as $E(i)$, related to the service requirement PoI i.*

Note that the UAV has to visit and provide service for all PoIs, so the total service energy consumption is written as:

$$E_M = \sum_{\forall i} E(i) \tag{9}$$

After explicitly decomposing the total energy consumption, we now recompose it by summing the costs of these parts discussed above, simply marked as E^+_{ALL}.

$$E^+_{ALL} = E_C + E_T + E_S + E_M. \tag{10}$$

Because the service requirements of PoIs are generally known before scheduling, the service energy consumption E_M is a known constant. We hence define the following partial energy consumption without the constant E_M.

$$E_{ALL} = E_C + E_T + E_S. \tag{11}$$

To minimize the total energy consumption E^+_{ALL} is equivalent to minimizing the partial energy consumption $E_{ALL} = E^+_{ALL} - E_M$, where E_M is a constant.

4.3 Problem formulation

Given the model described above, we are ready to define this problem as P1.

Definition 5 (P1)**.** *Given a set of PoIs and models mentioned above, the* general waypoint-based PoI-visiting *problem is to find a route for UAV to minimize the total energy consumption in Eq. (11), while the all PoIs visiting constraint Eq. (1), the range visiting constraint Eq. (2), the route connecting constraint Eq. (4) and Eq. (5) are satisfied.*

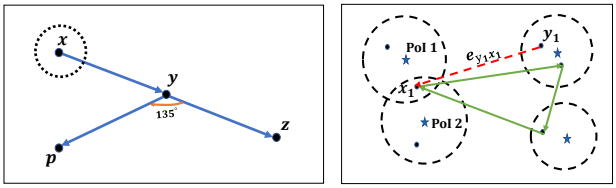
Formally, we give the following problem formulation:

$$\begin{aligned} \textbf{(P1)}\ & min\ Eq.\ (11) \\ & s.t.\ Eq.\ (1),\ (2),\ (4),\ (5), \\ & \quad Eq.\ (6),\ (7),\ (8). \end{aligned}$$

5 Graph Representation of Turning and Switching

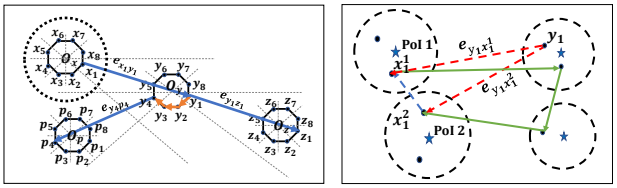
Problem P1 seeks a flight route, starting from PoI 0 and ending at PoI 0, selecting a series of waypoints to visit each PoI with the minimum total energy consumption in Eq. (11). This problem seems to have deep roots in classic graph problems, such as the generalized traveling salesman problem (GTSP). In GTSP, there is a set of cities and some subsets of these cities, where a salesman must visit every subset by one of its city with the shortest tour and ultimately return to the starting city. We can connect problem P1 to GTSP by mapping each PoI to one city subset, each waypoint candidate of any PoI to one city of a city subset, energy cost between waypoints to distance between cities. In this way, we map the major parts of P1 to the graph based GTSP.

However, one important gap is how to embed the turning cost and the switching cost into a graph, which are proportional



(a) (b)

Fig. 4. The energy cost of making turns and switching PoI-visiting in original problem. In (a), according to our real-world experiments, making turns also costs UAV energy, which is proportional to turning angles. In (b), switching visiting between PoIs also costs energy, which is proportional to the times of switching. These two part of energy consumption are both hard to be represented in a graph.



(a) (b)

Fig. 5. Modeling the energy cost of making turns and switching PoI-visiting in a graph. In (a), we utilize regular octagons to replace original waypoints, evenly dividing the infinite 360° turning angles into 8 ranges to present 8 heading directions of a UAV, where each corner of octagon corresponds to 45° direction range. The choice of other regular polygons is discussed in subsection 6.1. In (b), we separate the overlapping ranges by splitting waypoints into virtual copies, and assign the switching energy consumption of one switch to the weight of a connecting edge between two virtual copies of octagon vertices.

to the turning angle and the times of switching respectively. More specially, there are two difficulties with this problem: 1) the *turning energy consumption* in Definition 2 is unable to be intuitively reflected by any graph element such as the edge weight. For example in Fig. 4(a), at Waypoint y, its *turning energy consumption* is related to the turning angle, so its previous waypoint, x, and its next waypoint, z, directly affect the turning cost on the route. Both two waypoints can not be determined when the graph is constructed, so it is arduous to represent the turning cost at Waypoint y on the graph by edge weight. And 2) the *switching energy consumption* in Definition 3 can not be easily represented in the graph model either. For example in Fig. 4(b), Waypoint x_1 is in the overlap of PoI 1 and PoI 2, so the UAV can either visit (motionlessly) the two PoIs sequentially with switching cost at x_1, or only visits one PoI without switching cost. However, when the graph is constructed, the two possibilities are both open, so it is hard to be modeled by edge weight. Therefore, how to convert a) the energy consumption of straight flight, b) the energy consumption of turning, and c) the energy consumption of switching PoI-visiting, into a unified form (edge weight) on the graph model is our main task. In the following subsections, we propose a novel approach to solve this problem ingeniously.

We start with a simple case, *i.e.,* modeling energy cost of straight flight by graph $G_1(\mathbb{S}_1, \mathbb{E}_1, \mathbb{W}_1)$. The vertex set $\mathbb{S}_1$ is defined to encompass all waypoint candidates, $\mathbb{S}_1 = S_1 \cup S_2 \cup \cdots \cup S_n$. The edge set $\mathbb{E}_1$ includes directed edge e_{xy} if waypoint candidate x and y belong to different PoIs. In order to model the energy cost of straight flight from x to y on e_{xy}, we directly set the edge weight $W(e_{xy}) = E(e_{xy})$. And $\mathbb{W}_1$ includes all weights of edges. Hence, graph $G_1(\mathbb{S}_1, \mathbb{E}_1, \mathbb{W}_1)$ is generated.

5.1 Modeling energy cost of making turns

This subsection improves graph G_1 to include the energy cost of making turns, and generates graph G_2. According to our practical energy consumption model, we have the turning energy consumption $E(\theta) = c_2\theta + C_2$, where θ is the heading angle changed, and c_2 and C_2 are factors related to a special UAV. Since energy C_2 is constant for any turn, it can be added to the weight of edges directly, $W(e_{xy}) = E(e_{xy}) + C_2, \forall x, y$. Now we focus on modeling the proportional energy to angles $c_2\theta$.

The core idea is that we approximate the infinitely precise turning angle into a set of finite options, and use edge weight to represent the energy cost of making turns. To simplify the illustration and make it easier to calculate, we utilize regular octagons as drawn in Fig. 5(a) to replace original waypoints in Fig. 4(a). It is also feasible to choose other regular polygons, like hexagon, decagon and so on. We present detailed theoretical analysis on the choice of different regular polygons in the next section. For the choice of regular octagons, we evenly divide the infinite 360° turning angles into 8 ranges to present 8 heading directions of a UAV, where each corner corresponds to one direction range of 45°. Hence, when a UAV makes a turn, the change of its heading direction is demonstrated by two corners on the octagon, *i.e.,* the arrival corner and departure corner. In this way, the turning is proportional to the path weight sum between the two corners, as shown in Fig. 5(a). Furthermore, during a straight flight path, a UAV does not change its heading direction, so the departure corner of the starting octagon must match the arrival corner of the ending octagon. For this reason, the departure corner determines not only a direction range but also the reachable octagons in this direction. For example, waypoint candidate y in Fig. 4(a) can reach waypoint candidate z and p, while in Fig. 5(a), the octagon of y reaches the octagon of z and the octagon of p from different departure corners that represent different heading direction ranges.

Formally, we convert graph $G_1 = (\mathbb{S}_1, \mathbb{E}_1, \mathbb{W}_1)$ into $G_2 = (\mathbb{S}_2, \mathbb{E}_2, \mathbb{W}_2)$. Any vertex (waypoint) $x \in \mathbb{S}_1$, is expanded into a regular octagon, denoted as O_x, with 8 vertices, indexed as $x_1, x_2, \cdots, x_8$, and each range thus represents one 45° direction. Here we let $x_9 = x_1$ for loop purpose. The set $\mathbb{S}_2$ includes the vertices of all octagons. Between each two adjacent vertices on O_x, such as x_i and $x_j, |i-j| = 1$, there is an edge, $e_{x_ix_j} \in \mathbb{E}_2$, whose weight is set to $W(e_{x_ix_j}) = 45°c_2$, representing the cost of a 45° turn. For straight flight, such as from O_x to O_y, the UAV keeps its heading direction unchanged, which means the direction index of departure corner x_i on O_x must cover O_y, and the arrival corner on O_y has the same direction index as x_i, *i.e.,* y_i. We define the index of the flight direction from x to y as $d_{xy} = 1, 2, \cdots, 8$. Therefore, for any edge $e_{xy} \in \mathbb{E}_1$, we create an edge $e_{x_iy_i} \in \mathbb{E}_2$, where $i = d_{xy}$. Note that O_x is an infinitesimal regular octagon without physical significance, so the weight of $e_{x_iy_i}$ is equal to the weight of original edge e_{xy}, *i.e.,* $W(e_{x_iy_i}) = W(e_{xy})$. Let $\mathbb{W}_2$ covers weights of all edges. The pseudocode is in Algorithm `ModelingTurns`.

An example is given in Fig. 5(a) to help readers to get a better sense on how our modeling works. There are four waypoints x, y, z and p, thus we have four octagons O_x, O_y, O_z and O_p. First, we check the path $x \to y \to z$. Assume the UAV starts at direction 1 of O_x, *i.e.,* x_1. Since O_y is within the direction index of x_1, there is an edge $e_{x_1y_1}$, according to Line 8 of

Algorithm 1: ModelingTurns ($G_1(\mathbb{S}_1, \mathbb{E}_1, \mathbb{W}_1)$)

$\mathbb{S}_2 = \emptyset, \mathbb{E}_2 = \emptyset, \mathbb{W}_2 = \emptyset$;
for *each* $x \in \mathbb{S}_1$ **do**
  $\mathbb{S}_2 = \mathbb{S}_2 \cup \{x_1, x_2, \cdots, x_8\}$;
  $\mathbb{E}_2 = \mathbb{E}_2 \cup \{e_{x_i y_j} || i-j| = 1, i, j = 1, 2, \cdots, 9\}$;
  $\mathbb{W}_2 = \mathbb{W}_2 \cup \{W(e_{x_i x_j}) = 45°c_2\}, |i-j| = 1, i, j = 1, 2, \cdots, 9$;
end
for *each* $e_{xy} \in \mathbb{E}_1$ **do**
  $\mathbb{E}_2 = \mathbb{E}_2 \cup e_{x_i y_i}$, where $i = d_{xy}$;
  $\mathbb{W}_2 = \mathbb{W}_2 \cup \{W(e_{x_i y_j}) = W(e_{xy})\}$;
end
return $G_2(\mathbb{S}_2, \mathbb{E}_2, \mathbb{W}_2)$

Algorithm ModelingTurns. Then the UAV moves along $e_{x_1 y_1}$ to reach y_1. Next the UAV continues at y_1, and then goes through edge $e_{y_1 z_1}$ to arrive at z_1 by the same logic. Note that the energy consumed on acceleration/deceleration has been modeled on the weight of edge $e_{x_1 y_1}$ and $e_{y_1 z_1}$ already. Second, we check the path $x \to y \to p$, where the UAV has to make a turn at y. Similarly, assume the UAV starts at x_1 and arrives at y_1 through edge $e_{x_1 y_1}$. Subsequently, since O_p is outside the direction range of vertex y_1, at first the UAV chooses the octagonal edges, *i.e.,* $e_{y_1 y_2}$, $e_{y_2 y_3}$ and $e_{y_3 y_4}$, to make three 45° turns to arrive at y_4 according to the algorithm. Then it directly reaches the target p_4 through edge $e_{y_4 p_4}$. In this case, the energy cost of making three 45° turns can be calculated by the three-edges weight of octagon. In conclusion, our modeling of turns works correctly, the larger the turning angle, the more the sum of edge weight on the octagon.

5.2 Modeling energy cost of switching PoI-visiting

In this subsection, we pay special attention to the switching energy cost of PoI-visiting and generate a new graph G_3. By the definition of the energy cost of switching PoI-visiting, $E(D_x) = c_3(|D_x| - 1)$, if $|D_x| > 1$, there is a switching cost at x, which must be skillfully reflected by the edge of the graph.

Our core idea is that, for any waypoint candidate in the common area of multiple PoIs, we split it into virtual waypoint candidate copies, one for each PoI. There is an edge between any two copied vertices, and we can assign the cost of one switching to its weight. This idea is inspired by [51]. Then we convert graph $G_2 = (\mathbb{S}_2, \mathbb{E}_2, \mathbb{W}_2)$ into $G_3 = (\mathbb{S}_3, \mathbb{E}_3, \mathbb{W}_3)$. Since P_x is the set of PoIs that covers waypoint candidate x, *i.e.,* $P_x = \{k | x \in S_k\}$, any vertex $x_i \in \mathbb{S}_2$ is converted to $|P_x|$ copies. And if $|P_x| > 1$, we connect every two copies with an edge weighted c_3, representing one switching cost between the involved PoIs. We present these steps formally in Algorithm ModelingSwitch.

An example is given in Fig. 5(b) to clarify some key steps in Algorithm ModelingSwitch. Assume Waypoint x_1 locates in the overlapping range of PoI 1 and PoI 2. Following Algorithm ModelingSwitch, first separate the overlapping ranges by making two copied of x_1, *i.e.,* x_1^1 and x_1^2, and then assign them to PoI 1 and PoI 2 one for each, as displayed in Fig. 5(b). Then an edge is added to connect the two copied vertex (the blue dashed line in Fig. 5(b)), whose weight is equal to the energy cost of one switching PoI-visiting. Moreover, every edge with an endpoint inside the overlapping range, such as edge $e_{y_1 x_1}$, is replaced by two new edges (red dashed lines in Fig. 5(b)), $e_{y_1 x_1^1}$ and $e_{y_1 x_1^2}$, with the same weight as that of $e_{y_1 x_1}$. Hereby, any switching cost is now reflected on the graph, and the total switching cost can be easily represented by summing up the weights of all involved edges.

Algorithm 2: ModelingSwitch ($G_2(\mathbb{S}_2, \mathbb{E}_2, \mathbb{W}_2)$)

$\mathbb{S}_3 = \emptyset, \mathbb{E}_3 = \emptyset, \mathbb{W}_3 = \emptyset$;
for *each* $x_i \in \mathbb{S}_2$ **and** *each* $p \in P_x$ **do**
  $\mathbb{S}_3 = \mathbb{S}_3 \cup \{x_i^p\}$;
  for *each* $q \in P_x$ **and** $p \neq q$ **do**
    $\mathbb{E}_3 = \mathbb{E}_3 \cup \{e_{x_i^p x_i^q}\}$;
    $\mathbb{W}_3 = \mathbb{W}_3 \cup \{W(e_{x_i^p x_i^q}) = c_3\}$;
  end
end
for *each* $e_{x_i x_j} \in \mathbb{E}_2$ **and** $p \in P_x$ **do**
  $\mathbb{E}_3 = \mathbb{E}_3 \cup \{e_{x_i^p x_j^p}\}$;
  $\mathbb{W}_3 = \mathbb{W}_3 \cup \{W(e_{x_i^p x_j^p}) = W(e_{x_i x_j})\}$;
end
for *each* $e_{x_i y_i} \in \mathbb{E}_2$ **do**
  for *each* $p \in P_x$ **and** $q \in P_x$ **do**
    $\mathbb{E}_3 = \mathbb{E}_3 \cup \{e_{x_i^p y_i^q}\}$;
    $\mathbb{W}_3 = \mathbb{W}_3 \cup \{W(e_{x_i^p y_i^q}) = W(e_{x_i y_i})\}$;
  end
end
return $G_3(\mathbb{S}_3, \mathbb{E}_3, \mathbb{W}_3)$

5.3 Redefinition of the problem by a graph model

Since the turning cost and the switching cost are modeled by Algorithm ModelingTurns and ModelingSwitch in the previous subsections, we can redefine P1 by the generated graph $G_3(\mathbb{S}_3, \mathbb{E}_3, \mathbb{W}_3)$. Here we define a new directed weighted graph $D(\mathbb{S}, \mathbb{E}, \mathbb{W}, S')$, where the vertex set $\mathbb{S} = \mathbb{S}_3$, the edge set $\mathbb{E} = \mathbb{E}_3$, the weight set $\mathbb{W} = \mathbb{W}_3$. Besides, the set $S' = \{S'_1, S'_2, \cdots S'_n\}$, where $S'_p = \{x_i^p | x \in S_p, i = 1, 2, \cdots, 8\}$ is vertex set for PoI $p, p = 1, 2, \cdots n$. Note that, we have $S'_p \cap S'_q = \varnothing, \forall p \neq q$ by ModelingSwitch.

Definition 6 (P2)**.** *Given a directed weighted graph $D(\mathbb{S}, \mathbb{E}, \mathbb{W}, S')$, the graph-based* general waypoint-based PoI-visiting *problem is to find a feasible tour in D to visit each subset $S'_p \in S'$ once, while the sum of weights of all edges chosen is minimum.*

Since problem P1 can be converted into P2, step by step via intermediate graph model G1, G2, G3, D for transformation, now we are ready to explicate the direct correspondence between P1 and P2.

Each waypoint candidate x in P1, following the elaborate transformation of the modeling making turns in Section 5.1 and the modeling switching PoI-visiting in 5.2, has its corresponding vertices of octagon in graph D of P2, hence the vertex set $\mathbb{S}$ can be denoted as:

$$\mathbb{S} = \{x_i^p | x \in S, i = 1, 2, \cdots, 8, p \in P_x\}, \tag{12}$$

where S covers all waypoint candidates in P1, and P_x includes the PoI that covers waypoint candidate x.

Since we have learned the correspondence between waypoint candidates in P1 and vertices in the graph D of P2, the edges in D are easy to be determined. The edge set $\mathbb{E}$ covers three types of edges. a) The edge between adjacent vertices of an octagon, representing a 45° turning[1], $e_{x_i^p x_j^p}$, where $|i - j| =$

1. We set $x_9 = x_1$ for loop purpose.

$1, i, j = 1, 2, \cdots, 9$. b) The edge between the same-index vertices of different octagons, representing a straight flight between two different waypoint candidates in P1, $e_{x_i^p y_i^q}$, where $i = d_{xy}$ is the index of the flight direction from x to y. c) The edges between two virtual copies of octagon vertices, representing a switching PoI-visiting between their corresponding waypoint candidates in P1, $e_{x_i^p x_i^q}$. So $\mathbb{E}$ can be denoted as:

$$\begin{aligned}\mathbb{E} = \{e_{x_i^p x_j^p} | x \in S, p \in P_x, |i-j| = 1, i, j = 1, 2, \cdots, 9\} \cup \\ \{e_{x_i^p y_i^q} | x \in S, y \in S, p \in P_x, q \in P_y, i = d_{xy}\} \cup \\ \{e_{x_i^p x_i^q} | x \in S, p \in P_x, q \in P_x, i = 1, 2, \cdots, 8, p \neq q\}. \quad (13)\end{aligned}$$

By Definition 1, 2 and 3, we assign different weights to the edges in $\mathbb{E}$, corresponding to different types of UAV energy consumption in P1: the weights of the edges between adjacent vertices of octagon, set to the energy cost of making a 45° turn, $45c_2$; the weights of the edges between the same-index vertices of octagons, set to the energy cost of a straight flight between their corresponding waypoint candidates, $E(e_{xy}) + C_2$; the weights of the edges between two virtual copies of octagon vertices, set to the energy cost of a switching PoI-visiting between their corresponding waypoint candidates, C_3. So we have

$$\begin{aligned}\mathbb{W} = \{W(e_{x_i^p y_i^q}) = E(e_{xy}) + C_2\} \cup \\ \{W(e_{x_i^p x_j^p}) = 45c_2\} \cup \\ \{W(e_{x_i^p x_i^q}) = C_3\}. \quad (14)\end{aligned}$$

To sum up, we have clarified the direct correspondence between problem P1 and problem P2. But how the problem redefinition affects the solution is still unanswered, we thus give the theoretical analysis in the next section.

6 Theoretical Analysis and Solution

In this section, we first provide theoretical analysis for the graph redefinition, and then present the solution to the graph-based P2.

6.1 Theoretical analysis for the graph redefinition

The purpose of this subsection is to investigate how the graph redefinition from problem P1 to problem P2 affects the solutions. More specially, we provide theoretical analysis to see the difference between the value of any feasible solution of P1 and the value of the corresponding solution of P2.

Lemma 1. *Any feasible solution of problem P2 can be uniquely mapped into a feasible solution of problem P1, and vice versa.*

Proof. Since the transformation from a feasible solution of P1 to that of P2 has been discussed in Section 5 in detail, here we only show how to map a feasible solution of P2 to one of P1.

Following the definition of P2, a feasible tour in D must visit each subset by at least one of its vertices. If there are two or more vertices in one subset *e.g.,* S'_p, edge type as $e_{x_i^p x_j^p} \in \mathbb{S}, |i-j| = 1$ must be included in the tour, because it is the only type of edges that can connect two vertices within the same subset. Besides, between any two subsets S'_p and S'_q, where $p \neq q$, there are only two types of edges, *e.g.,* $e_{x_i^p y_i^q} \in \mathbb{S}$ or $e_{x_i^p x_i^q} \in \mathbb{S}$, so either of which is possible to be included in this tour.

According to Eq. (12), (13), and (14), in a solution of P2, if an edge $e_{x_i^p y_i^q} \in \mathbb{S}$ is included in the tour, the UAV has to fly straight from Waypoint x to y with an energy cost of $E(e_{xy}) + C_2$ in P1; if an edge $e_{x_i^p x_j^p} \in \mathbb{S}, |i-j| = 1$ is included in the tour, the UAV has to make a 45° turn at Waypoint x with an energy cost of $45c_2$ in P1; if an edge $e_{x_i^p x_i^q} \in \mathbb{S}$ is included in the tour, the UAV has to switch visiting from PoI p to q with an energy cost of c_3 in P1. And we can conclude that the total weights of the tour in P2 is equal to the total UAV energy consumption of the flight in P1.

Clearly, we give a comprehensible explanation about the mapping relation between the solution of P1 and that of P2. □

Let α and $|\alpha|$ represent a feasible flight planning tour of P1 and its energy consumption, and let β and $|\beta|$ represent a feasible solution of P2 and its sum of weights. Define a function, $p1to2(\alpha)$, to convert α to a corresponding solution of P2. Similarly, define $p2to1(\beta)$ to convert β to a corresponding solution of P1.

Based on the proof of Lemma 1, we have already learned that $|\beta| = |p2to1(\beta)|$, while the relationship between $|\alpha|$ and $|p1to2(\alpha)|$ has to be clarified. More specifically, we define $\frac{|p1to2(\alpha)|}{|\alpha|}$ as the graph redefinition approximation ratio and provide theoretical analysis on its upper bound.

Definition 7 (Graph Redefinition Approximation Ratio)**.** *Let α^{opt} be the optimal solution for an instance of P1, and then define* ***OPT***$=|\alpha^{opt}|$ *and* ***OPT****$=|p1to2(\alpha^{opt})|$. *Let $\beta = p1to2(\alpha^{opt})$ and $\alpha^{alg} = p2to1(\beta)$, and then define* ***ALG***$=|\alpha^{alg}|$. *So we have the* Graph Redefinition Approximation Ratio R_{red}*:*

$$R_{red} = \frac{\boldsymbol{OPT*}}{\boldsymbol{OPT}} = \frac{\boldsymbol{ALG}}{\boldsymbol{OPT}} = \frac{|\alpha^{alg}|}{|\alpha^{opt}|}.$$

For the theoretical analysis on the upper bound of the graph redefinition approximation ratio, we do not restrict on utilizing regular octagons to approximate the choices of a turning angle, but regular polygon with H edges and thus one direction range of the polygon can be denoted as $\Theta = \frac{360}{H}$. In practical, a UAV does not turn an arbitrary small angle, but a minimum real-world angle, which is denoted as δ.

Theorem 1. *The upper bound of the* Graph Redefinition Approximation Ratio *is* $\max\{\Theta/\delta, 2\}$, i.e., $R_{red} \leq \max\{\Theta/\delta, 2\}$.

Proof. For an instance of P1, there is an optimal solution α^{opt}, and a converted corresponding solution of P2, $\beta = p1to2(\alpha^{opt})$. And we have $|\beta| = |p2to1(\beta)|$ by the proof of Lemma 1. Clearly, to solve $R_{red} = \frac{|p1to2(\alpha^{opt})|}{|\alpha^{opt}|}$ is equivalent to solving the ratio between **ALG**$=|p2to1(\beta)| = |\alpha^{alg}|$ and **OPT**$=|\alpha^{opt}|$.

Formally, let $|\alpha^{opt}| = E_{ALL}^{opt}$ and $|\alpha^{alg}| = E_{ALL}^{alg}$, so $R_{red} = E_{ALL}^{alg}/E_{ALL}^{opt}$. According to Eq. (11), we have

$$\begin{aligned}E_{ALL}^{opt} = E_C^{opt} + E_T^{opt} + E_S^{opt}, \\ E_{ALL}^{alg} = E_C^{alg} + E_T^{alg} + E_S^{alg}.\end{aligned}$$

Following the graph-based transformation from P1 to P2, for the energy consumption of both straight flight E_C and switching E_S, there is no difference between α^{alg} and α^{opt}. So we have

$$E_C^{alg} = E_C^{opt}, E_S^{alg} = E_S^{opt}.$$

As a result we focus on the comparison of turning energy consumption, E_T^{opt} and E_T^{alg}. Combined with Definition 2, E_T^{opt} and E_T^{alg} can be respectively defined as:

$$\begin{aligned}E_T^{opt} = \sum_j (c_2\theta_j^{opt} + C_2) = \sum_j c_2\theta_j^{opt} + \sum_j C_2, \\ E_T^{alg} = \sum_j (c_2\theta_j^{alg} + C_2) = \sum_j c_2\theta_j^{alg} + \sum_j C_2,\end{aligned}$$

where θ_j^{opt} and θ_j^{alg} are the j-th turning angles for α^{opt} and α^{alg} respectively.

Observe that θ_j^{alg} and θ_j^{opt} are determined by their incoming and outgoing edges. For any incoming or outgoing angle θ, we have the following equation by Algorithm `ModelingTurns`,

$$\theta^{'} = \Theta(\lfloor\frac{\theta}{\Theta}\rfloor + \frac{1}{2}).$$

We hence can utilize the subtraction of incoming and outgoing angles to represent an angle, *i.e.,*

$$\theta_j^{opt} = \theta_1 - \theta_2,$$
$$\theta_j^{alg} = \theta_1^{'} - \theta_2^{'} = \Theta(\lfloor\frac{\theta_1}{\Theta}\rfloor + \frac{1}{2}) - \Theta(\lfloor\frac{\theta_2}{\Theta}\rfloor + \frac{1}{2}).$$

Without loss of generality, we assume $180^\circ \geq \theta_1 > \theta_2 \geq 0$. Clearly, we have

$$\begin{aligned} R_{red} =& \frac{E_T^{alg}}{E_T^{opt}} = \frac{E_C^{alg} + E_S^{alg} + \sum_j c_2\theta_j^{alg} + \sum_j C_2}{E_C^{opt} + E_S^{opt} + \sum_j c_2\theta_j^{opt} + \sum_j C_2} \\ <& \frac{\sum_j c_2\theta_j^{alg}}{\sum_j c_2\theta_j^{opt}} \leq \max_{\forall j}\{\frac{\theta_j^{alg}}{\theta_j^{opt}}\} \\ =& \max_{\forall\theta_1,\forall\theta_2}\{\frac{\Theta(\lfloor\frac{\theta_1}{\Theta}\rfloor + \frac{1}{2}) - \Theta(\lfloor\frac{\theta_2}{\Theta}\rfloor + \frac{1}{2})}{\theta_1 - \theta_2}\} \\ =& \max_{\forall\theta_1,\forall\theta_2}\{\frac{\lfloor\frac{\theta_1}{\Theta}\rfloor - \lfloor\frac{\theta_2}{\Theta}\rfloor}{\frac{\theta_1}{\Theta} - \frac{\theta_2}{\Theta}}\} = \max_{\forall d_1,\forall d_2}\{\frac{\lfloor d_1\rfloor - \lfloor d_2\rfloor}{d_1 - d_2}\}, \end{aligned}$$

where we set $d = \theta/\Theta$ for the last equation.

Actually, $\Theta\cdot(\lfloor d_1\rfloor - \lfloor d_2\rfloor)$ and $\Theta\cdot(d_1 - d_2)$ are the turning angles respectively derived from **ALG** and **OPT**. Now we analyze R_{red} by classification.

(i) $\Theta\cdot(\lfloor d_1\rfloor - \lfloor d_2\rfloor) = 0$, which means there is no turning, so

$$R_{red} \leq \max_{\forall d_1,\forall d_2}\{\frac{\lfloor d_1\rfloor - \lfloor d_2\rfloor}{d_1 - d_2}\} = 0$$

(ii) $\Theta\cdot(\lfloor d_1\rfloor - \lfloor d_2\rfloor) = \Theta$, implying that there is a turn with Θ degrees. Moreover, $\Theta\cdot(d_1 - d_2) = \theta_1 - \theta_2 \geq \delta$, so

$$R_{red} \leq \max_{\forall d_1,\forall d_2}\{\frac{\lfloor d_1\rfloor - \lfloor d_2\rfloor}{d_1 - d_2}\} \leq \frac{\Theta}{\delta}$$

(iii) $\Theta\cdot(\lfloor d_1\rfloor - \lfloor d_2\rfloor) = \Theta\cdot Q$, where $Q \geq 2, Q \in \mathbb{R}^+$, indicating there are Q Θ-degree turns. Then, we must have $\Theta\cdot(d_1 - d_2) \geq \Theta\cdot(Q-1)$, so

$$R_{red} \leq \max_{\forall d_1,\forall d_2}\{\frac{\lfloor d_1\rfloor - \lfloor d_2\rfloor}{d_1 - d_2}\} \leq \frac{Q}{Q-1} < 2$$

To sum up, we have proven $R_{red} \leq \max\{\Theta/\delta, 2\}$. In other words, the upper bound of the *Graph Redefinition Approximation Ratio* is $\max\{\Theta/\delta, 2\}$. □

A very large and loose upper bound does not have any theoretical significance, although it is technically correct. So, it is important to find a tight upper bound.

Theorem 2. $\max\{\Theta/\delta, 2\}$ *is a tight upper bound of the* Graph Redefinition Approximation Ratio R_{red}.

Proof. We prove this theorem is correct by giving an example whose graph redefinition approximation ratio R_{red} equals $\max\{\Theta/\delta, 2\}$.

We construct an optimal solution α^{opt} of problem P1 with the following properties: the whole cyclic tour is divided into $H = 360^\circ/\Theta$ segments, and each segment consists of m δ-degree turns and one Θ-degree turn in order. Let $\Theta = 45^\circ$, then a more specific illustration of this example is demonstrated in Fig. 6. We assume other energy consumption is ignorable compared to the proportional part of the turning energy consumption. Consequently, we calculate the energy consumption of **OPT**:

$$E^{opt} = E_T^{opt} = (H\cdot m\cdot\delta + 360^\circ)c_2.$$

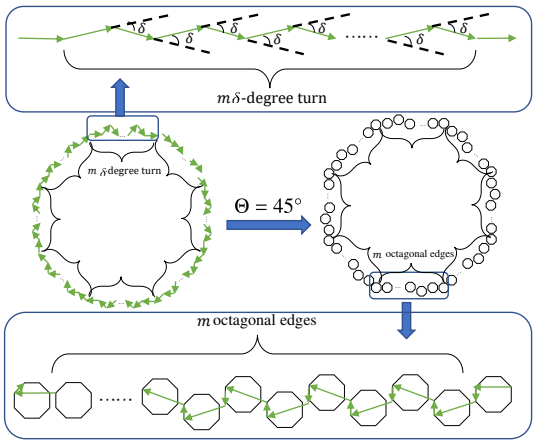


Fig. 6. A feasible tour of problem P1 and its corresponding tour of problem P2. The tour of P1 has of 8 segments, each of which consists of m δ-degree turns. By our proposed algorithm, each of these segments consists of m Θ-degree turns. If we let $\Theta = 45^\circ$, each segment thus consists of m octagonal edges as shown.

By carefully design the incoming angle and outgoing angle in the constructed example as in Fig. 6, each δ-degree turn is approximately represented by an edge in $\beta = p1to2(\alpha^{opt})$, where each edge is mapped into a Θ-degree turn in $\alpha^{alg} = p2to1(\beta)$. So, the sum of all these turning angles in **ALG** is $H\cdot m\cdot\Theta + 360^\circ$. Therefore, we have

$$E^{alg} = E_T^{alg} = (H\cdot m\cdot\Theta + 360^\circ)c_2.$$

Since $R_{red} = E_T^{alg}/E_T^{opt}$, we have

$$\lim_{m\to\infty} R_{red} = \frac{\Theta}{\delta}.$$

In short, with the elaboration of this example, we can prove the upper bound of the *Graph Redefinition Approximation Ratio*, $R_{red} = \max\{\Theta/\delta, 2\}$, is tight. □

Lemma 2. *The number of vertices and edges in the graph* $D(\mathbb{S}, \mathbb{E}, \mathbb{W}, S')$ *of problem P2 can be respectively denoted as* $F^{|\mathbb{S}|}(\Theta)$ *and* $F^{|\mathbb{E}|}(\Theta)$*:*

$$F^{|\mathbb{S}|}(\Theta) = \frac{360^\circ}{\Theta}\widehat{S},$$
$$F^{|\mathbb{E}|}(\Theta) = \frac{720^\circ}{\Theta}(2\widehat{S} - \widetilde{S}) + 2(\widehat{S}^2 - 2\widehat{S} + \widetilde{S}),$$

where

$$\widehat{S} = \sum_{\forall i}|S_i|, \quad \widetilde{S} = |\bigcup_{\forall i} S_i|,$$

S_i covers all waypoint candidates in PoI i, and $\widehat{S}$ and $\widetilde{S}$ are both constants for any given problem instance.

Proof. Based on the transformation in Section 5.1 and 5.2, each waypoint candidate in P1 has $H = 360^\circ/\Theta$ vertices in D, and

the number of polygons is the sum of waypoint candidates in each PoI, so clearly the number of vertices in D satisfies $F^{|\mathbb{S}|}(\Theta)$.

While the edge set of D consists of three parts.

(i) Edges inside polygons. The number of such directed edges is twice the number of all vertices in D, $2 \cdot F^{|\mathbb{S}|}(\Theta)$.

(ii) Edges between the copied polygons derived from the same waypoint candidate. For any two subsets, *e.g.,* S_i and S_j, following Section 5.2, the number of copied polygons is $|S_i \cap S_j|$, and there are $2 \cdot 360°/\Theta$ directed edges between each pair of the copied polygons, so the total number of this type of edges is:

$$\begin{aligned}&\sum_{\forall i,\forall j,i\neq j} 2 \cdot \frac{360°}{\Theta} \cdot |S_i \cap S_j| \\ =&\frac{720°}{\Theta}(\sum_{\forall i}|S_i| - |\bigcup_{\forall i} S_i|) = \frac{720°}{\Theta}(\widehat{S} - \widetilde{S}).\end{aligned}$$

(iii) Edges between the polygons derived from different waypoint candidates. For any two subsets, *e.g.,* S_i and S_j, the number of copied polygons is $|S_i \cap S_j|$. Based on Section 5.1, between S_i and S_j, there are $2(|S_i| \cdot |S_j| - |S_i \cap S_j|)$ directed edges, so the total number of this type of edges is:

$$\begin{aligned}&\sum_{\forall i,\forall j,i\neq j} 2(|S_i| \cdot |S_j| - |S_i \cap S_j|) \\ =&2(((\sum_{\forall i}|S_i|)^2 - \sum_{\forall i}|S_i|) - (\sum_{\forall i}|S_i| - |\bigcup_{\forall i} S_i|)) \\ =&2(\widehat{S}^2 - 2\widehat{S} + \widetilde{S}).\end{aligned}$$

To summarize, we add up the number of these three types edges in graph D, that is:

$$\begin{aligned}F^{|\mathbb{E}|}(\Theta) =&2 \cdot F^{|\mathbb{S}|}(\Theta) + \frac{720°}{\Theta}(\widehat{S} - \widetilde{S}) + 2(\widehat{S}^2 - 2\widehat{S} + \widetilde{S}) \\ =&\frac{720°}{\Theta}(2\widehat{S} - \widetilde{S}) + 2(\widehat{S}^2 - 2\widehat{S} + \widetilde{S}).\end{aligned}$$

□

Theorem 3. *The smaller the Graph Redefinition Approximation Ratio, the more vertices and edges in the graph D of problem P2.*

Proof. Based on Theorem 2, the Graph Redefinition Approximation Ratio has a tight upper bound, *i.e.,* $R_{red} < \Theta/\delta$ where Θ is one direction range of the polygon in the graph of problem P2 and δ is the minimum turning angle of UAV during its real-world flight. Clearly, Θ is proportional to R_{red} for a given δ. From Lemma 2, we learn that if Θ decreases, then both the number of vertices $F^{|\mathbb{S}|}(\Theta)$ and the number of edges $F^{|\mathbb{E}|}(\Theta)$ increase. Therefore, R_{red} and $F^{|\mathbb{S}|}(\Theta)/F^{|\mathbb{E}|}(\Theta)$ are negatively correlated. In other words, the smaller the Graph Redefinition Approximation Ratio, the more vertices and edges in the graph D. □

6.2 Solution to Problem P2

Following the definition of P2, the edge between vertex x_i^p and y_j^q is denoted as $e_{x_i^p y_j^q}$. Here we define the weight of $e_{x_i^p y_j^q} \in \mathbb{E}$ as $c_{x_i^p y_j^q}$, and $w_{x_i^p y_j^q}$ represents whether $e_{x_i^p y_j^q}$ is in the flight route,

$$w_{x_i^p y_j^q} = \begin{cases} 1, & \text{edge } e_{x_i^p y_j^q} \text{ is in the route,} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Subsequently, all vertices are divided into non-overlapping subsets, which are included by the set S' as:

$$S' = \{S'_p | p = 1, 2, \cdots, n\} \tag{16}$$

where any subset $S'_p = \{x_i^p | x \in S_p, i = 1, 2, \cdots, 8\}$.

The objective of P2 stated in Definition 6 is to find a feasible tour with minimum sum of weights in the graph D, equivalent to the objective of P1 stated in Definition 5 that to find a tour with constraints to minimize the total UAV energy consumption. Now we formulate P2 with the objective function and constraints:

$$\textbf{(P2)}\ min \sum_{x_i^p, y_j^q \in \mathbb{S}, e_{x_i^p y_j^q} \in \mathbb{E}} c_{x_i^p y_j^q} w_{x_i^p y_j^q}. \tag{17}$$

s.t.

$$\sum_{x_i^p \in \mathbb{S}, y_j^q \in \mathbb{S}, p\neq q} \sum_{e_{x_i^p y_j^q} \in \mathbb{E}} w_{x_i^p y_j^q} \geq 1, \textit{ for any subsets } S_p^{'} \tag{18}$$

$$\left.\begin{array}{l} \sum_{e_{x_i^p y_j^q} \in \mathbb{E}, p\neq q} w_{x_i^p y_j^q} \geq 1 \\ \sum_{e_{y_j^q x_i^p} \in \mathbb{E}, p\neq q} w_{y_j^q x_i^p} \geq 1 \end{array}\right\} \textit{for all subsets } S_p^{'}. \tag{19}$$

$$\sum_{e_{x_i^p y_j^q} \in \mathbb{E}, p\neq q} w_{x_i^p y_j^q} - \sum_{e_{y_j^q z_k^f} \in \mathbb{E}, q\neq f} w_{y_j^q z_k^f} = 0$$
$$\textit{for all vertices } y_j^q \in \mathbb{S}. \tag{20}$$

$$\sum_{x_i^p, S_p \in G} \sum_{y_i^q, S_q \notin G} \sum_{e_{x_i^p y_j^q} \in \mathbb{E}} w_{x_i^p y_j^q} \geq 1$$
$$\textit{for all sets G which are subsets of the collection of set } \mathbb{S},$$
$$2 \leq |G| \leq n - 2. \tag{21}$$

$$w_{x_i^p y_j^q} \in \{0, 1\} \textit{ for all } e_{x_i^p y_j^q} \in \mathbb{E}. \tag{22}$$

6.2.1 Constraints and transformation to GTSP

To facilitate solving the problem P2, we intend to transform it into the GTSP, which is defined as follows:

Definition 8 (GTSP)**.** *[22] Given a complete weighted graph $G = (V, E, w)$ on n vertices and a partition of V into m sets $P_V = \{V_1, ..., V_m\}$, where $V_i \cap V_j = \emptyset$ for all $i \neq j$ and $U_{i=1}^m V_i = V_j$, find a cycle in G that contains exactly one vertex from each set $V_i, i \in 1, ..., m$ and has minimum length.*

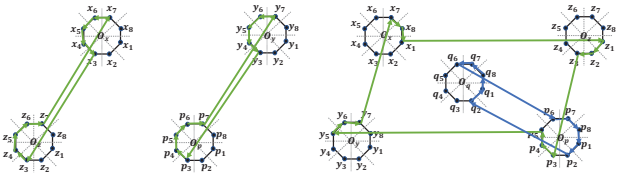
There are three constraints imposed to equalize P2 and GTSP:

- **Subset coverage**. Each subset must be visited at least once, which means in-edge and out-edge both necessarily exist in each subset. Eq. (19) unfolds this constraint.
- **Tour continuity**. Each vertex has the same in-degree as the out-degree to keep the tour continuous. We can use Eq. (20) to guarantee the continuity.
- **Subloop avoidance**. As shown in Fig. 7(a), the tour is impracticable due to the possible subloops. The constraint as Eq. (21) is crucial to avoid this case.

However, our solution is not rigorous enough. As shown in Fig. 7(b), a tour complies the three constraints but is infeasible (a subloop is marked in blue). To fix this little bug, we modify the constraint in Eq. (19) as follows:

$$\left.\begin{array}{l} \sum_{e_{x_i^p y_j^q} \in \mathbb{E}, p\neq q} w_{x_i^p y_j^q} = 1 \\ \sum_{e_{y_j^q x_i^p} \in \mathbb{E}, p\neq q} w_{y_j^q x_i^p} = 1 \end{array}\right\} \textit{for all subsets } S_p^{'} \tag{23}$$

where each subset is visited once and only once.



(a) (b)

Fig. 7. Two practical cases of subloop in a flight route.

According to the modeling of making turns in Section 5.1, the edges are formed only between two vertices in the same-index range. To get closer to GTSP, we add an extra operation to make the graph a complete one: for any two vertices without connection, assign them an edge, whose weight is equals to the sum of cost from one vertex to another. For example, two vertices in different regular octagons without connection, *i.e.,* x_i^p and x_j^q, and we can create an edge x_{ij}^{pq}, whose weight is the sum of the weights of the involved edges from x_i^p to x_j^q.

6.2.2 Large Neighborhood Search based algorithm usage

Now we have converted the original problem to GTSP and there always exists a feasible tour. Then we can expediently address our problem by referring to the Large Neighborhood Search based algorithm [52], where authors provide an effective heuristic library to solve GTSP efficiently. This library is based on adaptive large neighborhood search, mainly by iteratively removing and inserting vertices, to find a well-performing tour.

6.2.3 Time complexity of the proposed algorithm

Recall that an $N \times M$ size region is discretized into multiple $g \times g$ size grids and a grid is represented by its grid center, called a *waypoint candidate*. From Lemma 2, we have $\widetilde{S} = |\bigcup_{\forall i} S_i|$, and $\widehat{S} = \sum_{\forall i} |S_i|$, where S_i is the set of waypoint candidates covered by PoI i. The following lemma is to analyze $\widetilde{S}$ and $\widehat{S}$.

Lemma 3.

$$\widetilde{S} = O(\frac{NM}{g^2}), \widehat{S} = O(\frac{NMn}{g^2}).$$

Proof. Since the total number of waypoint candidates in this region is $\frac{NM}{g^2}$, which is larger than $|\bigcup_{\forall i} S_i|$, so $\widetilde{S} = |\bigcup_{\forall i} S_i| \leq \frac{NM}{g^2}$. Since for each S_i, we have $|S_i| \leq \frac{R_i}{g^2}$, where $|R_i|$ is the area covered by the range of PoI i. Hence, $\widehat{S} = \sum_{\forall i} |S_i| \leq \sum_{\forall i} \frac{|R_i|}{g^2} \leq \frac{NMn}{g^2}$. So we finally have $\widetilde{S} = O(\frac{NM}{g^2})$ and $\widehat{S} = O(\frac{NMn}{g^2})$. □

According to Lemma 3, it takes $O(\frac{NM}{g^2})$ steps to finish the discretization, and obtain all the waypoint candidates. To extend each waypoint candidate into an octagon and split the overlapping ones, it needs $O(\frac{NMn}{g^2})$ steps. Then generating edges and assigning weight to them takes $O(\widehat{S}^2)$ steps based on Lemma 2. Finally, we obtain an accessible graph model and refer to algorithm [52] to address it which takes $O(min\{n\widehat{S}^2, n^3\widehat{S}\log n\})$ time. In short, the proposed algorithm thus takes $O(min\{n\widehat{S}^2, n^3\widehat{S}\log n\})$ time in total.

7 Simulation

In this section, we implement our proposed flight planning algorithm, called the Optimization of Minimum-Energy by Graph Algorithm (OMEGA). And then conduct simulations to evaluate the performance of OMEGA. We use a brute-force method, the Enumerated Optimal Algorithm (EOA), to search for optimal solutions. We take the running time of EOA as a criterion to distinguish small scale problems from large scale problems, *i.e.,* if an optimal solution can be computed by EOA within 500 seconds, it is a small scale problem; otherwise, it is a large scale problem. For large scale problems, the Naive Minimum-Energy Algorithm (NMEA) is used for comparison. To show the energy efficiency of OMEGA, we add other two variants of OMEGA, denoted as O1 and O2 respectively, to specify the impact of the key energy components on the total energy consumption. Moreover, we also evaluate OMEGA under real-world settings.

7.1 Simulation settings

In our simulation, we code all algorithms by Python3.6. And simulations are conducted by using off-the-shelf desktop computer equipped with an Intel(R) Core(TM) i7-8700 CPU @3.20GHz and 16GB RAM running Window10 Professional Edition computing platform. The detailed simulation parameters are introduced as follows.

TABLE 2
Simulation parameters

Parameters \ Values	Values of small scale		Values of large scale	
	default	varying	default	varying
Side length of region	16	10 to 22	34	22 to 46
Number of PoIs	6	3 to 8	17	11 to 23
Number of waypoints in overlap	[6,7]	[0,1] to [10,11]	[14,15]	[8,9] to [20,21]
Radius of the range PoI	2.3	1.5 to 3.9	2.7	1.9 to 4.3
Grid granularity	1.5 unit			
Straight flight cost	120J each unit of length			
Turning cost	7.64J each degree			
Switching cost	900J each time			

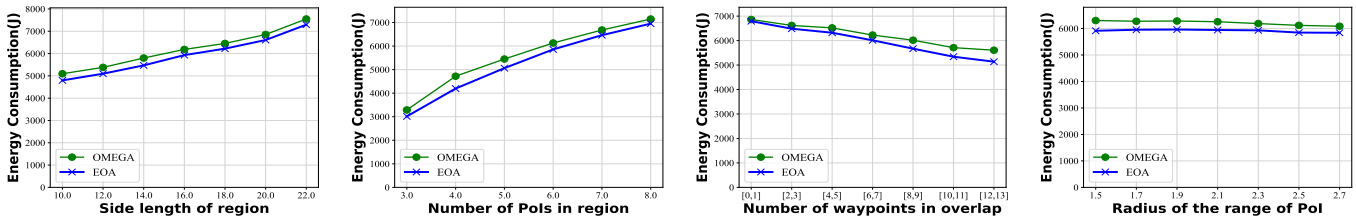
Based on our energy model of UAV flight introduced in the previous section, we set the straight flight cost to $120J$ each unit of length, the turning cost to $7.64J$ each degree, and the switching cost to $900J$ each time. We discretize the region by setting the grid granularity $g = 1.5$ unit. In the simulation, we choose the following main parameters to study their impact: the number of PoIs, n; the size of region, $N \times N$; the range of the number of the waypoint candidates in overlap, m; and the radius of the range of PoI, r. To evaluate their impact on algorithm performance, we focus on one parameter at a time, by varying its value and meanwhile fix the other parameters. We set the default value for n, N, m, r, *i.e.,* when the problem is small scale, set $n = 6$, $N = 16$, $m = [6, 7]$, $r = 2.3$; when the problem is large scale, set $n = 17$, $N = 34$, $m = [14, 15]$, $r = 2.7$. Table 2 lists the main parameters used in simulation. Unless otherwise specified, these parameters will be adopted in the default setting.

Then we run 100 times in each setting and take the average result of the 100 instances.

7.2 Results and discussion

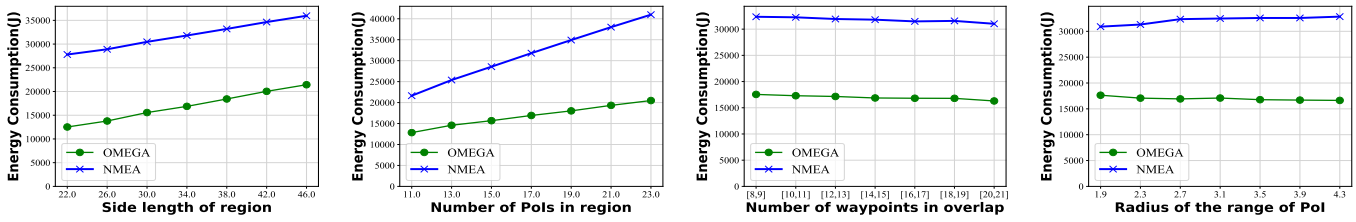
7.2.1 Algorithm comparison for small scale problems

When the problem scale is small, we use EOA to get the optimal result by enumerating all possible paths. The comparison between our OMEGA and EOA is shown in Fig. 8. As shown in Fig. 8(a), for both OMEGA and EOA, the longer the side length of region, the more the UAV energy consumption. Because with a fixed number of PoIs, the larger region leads to the longer distance,



(a) (b) (c) (d)

Fig. 8. Algorithm performance comparison for small scale problems. In (a), the longer the side length of region, the more energy consumption. In (b), the more number of PoIs, the more energy consumption. In (c), the energy consumption decreases as the number of waypoints in overlap increases. In (d), the energy consumption shows a slight decrease when the radius of the range of PoI becomes longer. In all these subplots, the result of OMEGA is close to that of EOA.



(a) (b) (c) (d)

Fig. 9. Algorithm performance comparison for large scale problems. In (a), for both OMEGA and NMEA, the longer the side length of region, the more energy consumption. In (b), for both OMEGA and NMEA, the more number of PoIs, the more energy consumption. In (c), for both OMEGA and NMEA, the energy consumption decreases as the number of waypoints in overlap increases. In (d), for OMEGA, the energy consumption shows a slight decrease when the radius of the range of PoI becomes longer, but for NMEA, the energy consumption shows a slight increase instead when the radius change in the same way. In all these subplots, compared to NMEA, OMEGA save almost half of energy consumption.

indicating the UAV needs more energy to visit PoIs. In Fig. 8(b), as the number of PoIs increases, energy cost increases. Since the UAV needs to visit more PoIs, more energy is consumed. In Fig. 8(c), we can see that the number of waypoint candidates in overlap increases while the energy cost decreases. According to the parameter setting of simulation, we require an increase in the number of waypoint candidates in overlap, and we fix the size of region, the number of PoI and the radius of the range of PoI, so the PoIs prefer to get closer to generate more waypoints in overlap during the position randomization process, which makes flight distance between PoI-visiting shorter to decrease the energy consumption. Furthermore, thanks to the intelligent strategy of our OMEGA, when there are more waypoint candidates in overlap which means more optional routes, *i.e.,* turning or switching PoI-visiting at overlapping waypoints, it is more likely to get the best one that has the minimum energy cost. In Fig. 8(d), longer radius length results in a slight decrease of energy consumption. The main reason for this tread is that the overlapping area becomes larger as the radius becomes longer, while the range of the number of total waypoint candidates is constant, which implies most waypoint candidates in region are in overlap. According to Fig. 8(c), we can understand that more candidate overlapping-waypoints is more likely to derive the optimal result to some extent. Obviously, the varying tendencies of the two curves are similar generally in these subplots. To be specific, the performance of our OMEGA is close to that of EOA on the whole, whose error is no more than 107%. And there are instances of feasible tours generated by OMEGA and EOA respectively in Fig. 10, whose tours are similar.

Furthermore, we compare the execution time of OMEAG and EOA in these four aspects, and the indicative numerical results are listed in Table 3. Clearly, in terms of efficiency, OMEGA outperforms EOA from these four subtables. Specially, when the radius of the PoI range $r = 3.9$, the execution time ratio between EOA and OMEGA is about $\frac{1005.9}{6.43} \approx 156$; when the number of PoI $n = 8$, the execution time ratio between EOA and OMEGA is even up to $\frac{11756.00}{2.5} \approx 5734$, which indicates the efficient performance of OMEGA compared with EOA.

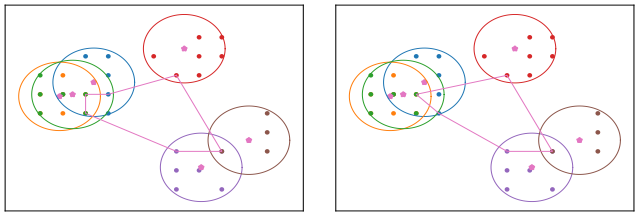
TABLE 3
Algorithm execution time comparison for small scale problems with variables: the side length of region N, the number of PoIs n, the radius of the PoI range r, and the number of waypoints in overlap $\mathcal{I}$

N	OMEGA(s)	EOA(s)
10	0.69	7.03
12	0.69	8.3
14	0.69	8.99
16	0.77	11.48
18	0.86	12.86
20	0.80	11.32
22	0.87	14.12

n	OMEGA(s)	EOA(s)
3	0.27	0.07
4	0.42	0.25
5	0.64	0.99
6	0.92	14.28
7	1.07	306.09
8	2.05	11756.00

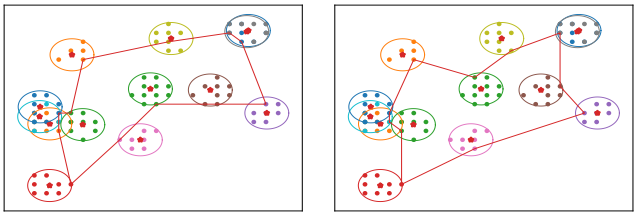
r	OMEGA(s)	EOA(s)
1.5	0.21	0.45
1.9	0.36	2.15
2.3	0.77	11.48
2.7	1.63	47.40
3.1	3.09	195.07
3.5	4.47	442.50
3.9	6.43	1005.90

$\mathcal{I}$	OMEGA(s)	EOA(s)
[0,1]	0.73	8.85
[2,3]	0.71	9.66
[4,5]	0.84	11.86
[6,7]	0.87	12.84
[8,9]	0.98	15.74
[10,11]	1.07	17.79
[12,13]	1.17	20.83



(a) (b)

Fig. 10. An example of the feasible tours in a small scale problem. The tours in (a) and (b) are generated by OMEGA, EOA respectively, and the two tours are almost coincident in the flight distance, the angle of turning and the number of PoI-switching waypoints.



(a) (b)

Fig. 11. An example of the feasible tours in a large scale problem. The tours in (a) and (b) are generated by OMEGA, NMEA respectively. The two tours are similar in trajectory (flight distance) while different in the angle of turning and the times of PoI-switching waypoints, resulting in widely varying energy consumption.

7.2.2 Algorithm comparison for large scale problems

When the problem scale is large, it is impossible to enumerate all routes due to the enormous time complexity. Hence, we verify the efficiency of OMEGA by comparing it with the NMEA where only the covered flight distance is considered. The comparison between OMEGA and NMEA is shown in Fig. 9. The analysis of the first three subplots of Fig. 9 gives similar conclusions to those in Fig. 8. For Fig. 9(a), the longer size length of region, the more energy consumption. For Fig. 9(b), the more number of PoIs in the region, the more energy consumption. For Fig. 9(c), with the increment of the number of overlapping-waypoints, the energy cost is less. However, for Fig. 9(d), the energy consumption of OMEGA is declining normally while that of NMEA is rising slightly. One explanation for the trend of NMEA is that most waypoint candidates are in overlap due to the expansion of the overlapping area, and the energy cost of NMEA is calculated without the restriction of switching PoI-visiting cost, so the UAV prefers to pick these waypoint candidates in overlap without switching PoI-visiting cost. Whereas, at last we have to calibrate this type of energy cost by adding the switching PoI-visiting cost that is not computed during the process of NMEA, so the total energy cost of NMEA increases inevitably. In short, compared with NMEA, the high-efficient OMEGA can save nearly 50% of energy consumption. And there are examples of feasible tours generated by OMEGA and NMEA respectively in Fig. 11, whose tours differ significantly.

7.2.3 Algorithm comparison between OMEGA, NMEA, O1 and O2

In previous subsections, it is clear that OMEGA has excellent performance. To show the energy efficiency of OMEGA more specifically, we compare OMEGA, NMEA and other two variants of OMEGA, O1 and O2.

OMEGA without switching PoI-visiting cost restrict(O1): We ignore UAV switching PoI-visitingt cost on the basis of OMEGA. In this variant, the total energy consumption is minimized only by straight flight distance and turning angle. According to the comparison result of OMEGA and O1, we can understand the impact of switching PoI-visiting cost on the total energy cost.

OMEGA without turning cost restrict(O2): In the similar way, we skip the turning cost when implementing OMEGA. This approach optimizes the total energy cost by weighing the straight flight distance and switching PoI-visiting cost. Then we can perceive the effect of turning cost by the difference of result between OMEGA and O2.

We compute the energy consumption for OMEGA, NMEA, O1, and O2 respectively, and count the flight distance, turning angle, and the time of switching PoI-visiting for each algorithm. The detailed statistics are listed in Table 4.

TABLE 4
Simulation Statistics

Components \ *Models*	OMEGA	NMEA	O1	O2
Energy consumption	39627.41	60499.79	42556.27	64454.21
Flight distance	119.22	113.88	114.72	117.42
Angle of turning	1144.8	990	910.8	1438.2
Times of PoI-switching	0.47	6.39	7.21	0.07

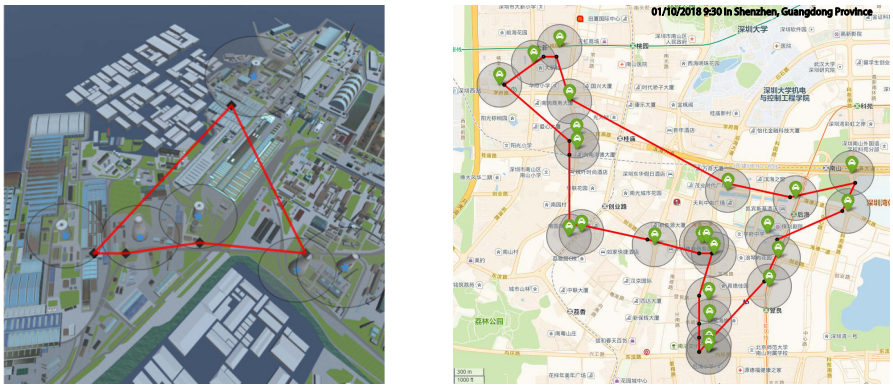
By comparing the results of these four solutions listed in Table 4, we can see that NMEA has the shortest flight distance, O1 gets the least tuning angle, and O2 gets the least times of switching PoI-visiting. While our OMEGA achieves the minimum energy consumption among these four solutions by tactfully handling the tradeoff among straight flight, turning and switching PoI-visiting for UAVs. Another interesting conclusion from Table 4 is that O2 unexpectedly consumes more energy than NMEA. We calculate the results of O2 and NMEA, and discover that O2 uses 448.2° turning and 3.54 unit of flight distance, just reduces 6.32 times of switching PoI-visiting. This implies that switching PoI-visiting cost is indispensable in this scenario.

7.3 OMEGA Flight planning in real scenarios

To better reflect the effectiveness of OMEGA in real scenarios, we make simulations with real-world datasets both in small scale and large scale.

The Fig. 12(a) demonstrates a small scale scenario example, whose background is a factory plant of Nanjing Iron and Steel Group, given several key facilities of the factory, *e.g.,* steel furnaces and chimneys. A UAV is dispatched to periodically monitor these facilities to prevent unexpected production situations, *e.g.,* abnormal temperature, humidity or pressure. In our simulation setting, there are 6 key monitor objects in factory, marked as blue stars in Fig. 12(a). An efficient-energy UAV trajectory to execute monitor tasks is devised by OMEGA as the line depicted in Fig. 12(a). From the flight planning path, it is obvious that the waypoints in overlapping areas of ranges will also be picked under the UAV energy consumption tradeoff.

The large scale scenario example is shown in Fig. 12(b), we utilize an agile and flexible UAV to provide edge computing timely for ground mobile users, *e.g.,* taxis or buses. Here we conduct simulations with the taxi dataset in [53], which covers the



(a) Chimneys patrol in Nanjing Iron and Steel Group, China.

(b) UAV-aided edge computing for taxi in Shenzhen, China.

Fig. 12. The effect of OMEGA flight planning in real scenario. The tour in (a) is designed for UAV to monitor several key steel furnaces and chimneys inside factory, while in (b) it shows the UAV flight planning path for serving many ground taxis with edge computing at a given moment.

detailed items of taxis, including the plate number, GPS latitude and longitude, and operation time, from October 1, 2018 0:00 to 24:00 in Shenzhen, Guangdong. We first filter and extract key information to determine the proper positions, operation time and the ranges of each computing offloading task, then by OMEGA we can get a more practical and efficient-energy tour for UAV to serve to ground mobile users as the red line plotted in Fig. 12(b).

8 Conclusion

In this paper, we formulate a general problem to match more application scenarios of UAVs, and we propose the *general waypoint-based PoI-visiting problem*. With the investigation of related work, most existing flight models simplify the UAV energy consumption, motivating us to build a more practical and accurate one by a set of real-world experiments. To address this energy minimization problem, we propose a novel graph-based energy-efficient approach, utilizing a well-studied classic solution of GTSP to find a tour with the minimum cost. Our theoretical analysis provides the tight upper bound for the graph redefinition approximation ratio. We conduct simulations by comparing with the best and the naive baseline respectively, to evaluate the performance of OMEGA. The final result shows that OMEGA is excellent-performance within 107% of the best result and nearly 50% of energy compared to the result of a naive algorithm.

Acknowledgement

This work was supported by the National Key R&D Program of China Grant 2021YFB2900100, the National Natural Science Foundation of China under grants 62072101, 62172091, 61872079, 62072102, 62132009, and 61632008, Jiangsu Provincial Key Laboratory of Network and Information Security Grant BM2003201, and Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China Grant 93K-9, partially supported by the Fundamental Research Funds for the Central Universities.

References

[1] J. Gong, T. Chang, C. Shen, and X. Chen, "Flight time minimization of UAV for data collection over wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1942–1954, 2018.

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

[3] J. Lyu, Y. Zeng, and R. Zhang, "Cyclical multiple access in UAV-aided communications: A throughput-delay tradeoff," *IEEE Wirel. Commun. Lett.*, vol. 5, no. 6, pp. 600–603, 2016.

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

[5] M. Chen, W. Liang, and Y. Li, "Data collection maximization for UAV-enabled wireless sensor networks," in *29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–9.

[6] Y. Li, W. Liang, W. Xu, Z. Xu, X. Jia, Y. Xu, and H. Kan, "Data collection maximization in IoT-sensor networks via an energy-constrained UAV," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2021.

[7] C. Luo, M. N. Satpute, D. Li, Y. Wang, W. Chen, and W. Wu, "Fine-grained trajectory optimization of multiple UAVs for efficient data gathering from WSNs," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 162–175, 2021.

[8] J. Zhang, Z. Li, W. Xu, J. Peng, W. Liang, Z. Xu, X. Ren, and X. Jia, "Minimizing the number of deployed UAVs for delay-bounded data collection of IoT devices," in *40th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2021, pp. 1–10.

[9] D. Vallejo, J. J. Castro-Schez, C. Glez-Morcillo, and J. Albusac, "Multi-agent architecture for information retrieval and intelligent monitoring by UAVs in known environments affected by catastrophes," *Eng. Appl. Artif. Intell.*, vol. 87, 2020.

[10] A. Khochare, Y. Simmhan, F. B. Sorbelli, and S. K. Das, "Heuristic algorithms for co-scheduling of edge analytics and routes for UAV fleet missions," in *40th IEEE Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[11] Q. Guo, J. Peng, W. Xu, W. Liang, X. Jia, Z. Xu, Y. Yang, and M. Wang, "Minimizing the longest tour time among a fleet of UAVs for disaster area surveillance," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2020.

[12] Y. Tang, Y. Miao, A. Barnawi, B. A. Alzahrani, R. Alotaibi, and K. Hwang, "A joint global and local path planning optimization for UAV task scheduling towards crowd air monitoring," *Comput. Networks*, vol. 193, p. 107913, 2021.

[13] S. Hosseinalipour, A. Rahmati, D. Y. Eun, and H. Dai, "Energy-aware stochastic UAV-assisted surveillance," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 5, pp. 2820–2837, 2021.

[14] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, 2019.

[15] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-efficient resource management in UAV-assisted mobile edge computing," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 249–253, 2021.

[16] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-

assisted mobile edge computing," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2021.
[17] X. Zhang and L. Duan, "Fast deployment of UAV networks for optimal wireless coverage," *IEEE Trans. Mob. Comput.*, vol. 18, no. 3, pp. 588–601, 2019.
[18] W. Chen, Z. Su, Q. Xu, T. H. Luan, and R. Li, "VFC-based cooperative UAV computation task offloading for post-disaster rescue," in *39th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 228–236.
[19] H. Huang and A. V. Savkin, "Viable path planning for data collection robots in a sensing field with obstacles," *Comput. Commun.*, vol. 111, pp. 84–96, 2017.
[20] "Mission Planning - Copter documentation," https://ardupilot.org/copter/docs/common-mission-planning.html, accessed Oct. 16, 2021.
[21] "Onboard SDK - DJI Developer," https://developer.dji.com/cn/onboard-sdk, accessed Oct. 16, 2021.
[22] S. Piao, Z. Ba, L. Su, D. Koutsonikolas, S. Li, and K. Ren, "Automating CSI measurement with UAVs: from problem formulation to energy-optimal solution," in *2019 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 2404–2412.
[23] C. Wang, F. Ma, J. Yan, D. De, and S. K. Das, "Efficient aerial data collection with UAV in large-scale wireless sensor networks," *IJDSN*, vol. 11, pp. 286 080:1–286 080:19, 2015.
[24] S. Ahmed, A. Mohamed, K. A. Harras, M. Kholief, and S. Mesbah, "Energy efficient path planning techniques for UAV-based systems with space discretization," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2016, pp. 1–6.
[25] Z. Huang, W. Wu, F. Shan, Y. Bian, K. Lu, Z. Li, J. Wang, and J. Wang, "Couas: Enable cooperation for unmanned aerial systems," *ACM Trans. Sens. Networks*, vol. 16, no. 3, pp. 24:1–24:19, 2020.
[26] F. Shan, J. Luo, R. Xiong, W. Wu, and J. Li, "Looking before crossing: An optimal algorithm to minimize UAV energy by speed scheduling with a practical flight energy model," in *39th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 1758–1767.
[27] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC planner: Energy efficient coverage path planning with multiple drones." IEEE, 2017, pp. 6182–6189.
[28] K. Goss, R. Musmeci, and S. Silvestri, "Realistic models for characterizing the performance of unmanned aerial vehicles," in *26th International Conference on Computer Communication and Networks, ICCCN 2017, Vancouver, BC, Canada, July 31 - Aug. 3, 2017*. IEEE, 2017, pp. 1–9. [Online]. Available: https://doi.org/10.1109/ICCCN.2017.8038444
[29] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Empirical power consumption model for uavs," in *88th IEEE Vehicular Technology Conference, VTC Fall 2018, Chicago, IL, USA, August 27-30, 2018*. IEEE, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/VTCFall.2018.8690666
[30] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, 2016.
[31] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 3, pp. 2109–2121, 2018.
[32] C. Zhan and Y. Zeng, "Aerial–ground cost tradeoff for multi-UAV-enabled data collection in wireless sensor networks," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1937–1950, 2020.
[33] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1208–1214.
[34] "QoS-aware UAV coverage path planning in 5G mmWave network," *Computer Networks*, vol. 175, p. 107207, 2020.
[35] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1262–1276, 2019.
[36] X. Wang, S. Garg, H. Lin, G. Kaddoum, J. Hu, and M. F. Alhamid, "An intelligent UAV based data aggregation algorithm for 5G-enabled internet of things," *Comput. Networks*, vol. 185, p. 107628, 2021.
[37] P. A. Apostolopoulos, M. Torres, and E. E. Tsiropoulou, "Satisfaction-aware data offloading in surveillance systems," ser. CHANTS'19. Association for Computing Machinery, 2019.
[38] M. Samir, D. Ebrahimi, C. Assi, S. Sharafeddine, and A. Ghrayeb, "Leveraging UAVs for coverage in cell-free vehicular networks: A deep reinforcement learning approach," *IEEE Trans. Mob. Comput.*, vol. 20, no. 9, pp. 2835–2847, 2021.
[39] J. Akshya and P. Priyadarsini, "Graph-based path planning for intelligent UAVs in area coverage applications," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 6, pp. 8191–8203, 2020.
[40] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, 2018.
[41] L. Zhang, A. Celik, S. Dang, and B. Shihada, "Energy-efficient trajectory optimization for UAV-assisted iot networks," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2021.
[42] M. B. Ghorbel, D. Rodriguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, "Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2165–2175, 2019.
[43] X. Xu, H. Zhao, H. Yao, and S. Wang, "A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2431–2443, 2021.
[44] Y. Li, Y. Fang, and L. Qiu, "Joint computation offloading and communication design for secure UAV-enabled MEC systems," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.
[45] D. Wei, J. Ma, L. Luo, Y. Wang, L. He, and X. Li, "Computation offloading over multi-UAV mec network: A distributed deep reinforcement learning approach," *Computer Networks*, vol. 199, p. 108439, 2021.
[46] R. Xiong and F. Shan, "Dronetank: Planning UAVs' flights and sensors' data transmission under energy constraints," *Sensors*, vol. 18, no. 9, p. 2913, 2018.
[47] F. Morbidi, R. Cano, and D. Lara, "Minimum-energy path generation for a quadrotor UAV," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1492–1498.
[48] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 6, pp. 3747–3760, 2017.
[49] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 4, pp. 2329–2345, 2019.
[50] J. Huang, F. Shan, R. Xiong, Y. Shao, and J. Luo, "Energy-efficient uav flight planning for a general poi-visiting problem with a practical energy model," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–10.
[51] C. Noon and J. Bean, "An efficient transformation of the generalized traveling salesman problem," *Information Systems and Operational Research*, vol. 31, 02 1993.
[52] S. L. Smith and F. Imeson, "GLNS: an effective large neighborhood search heuristic for the generalized traveling salesman problem," *Comput. Oper. Res.*, vol. 87, pp. 1–19, 2017.
[53] "Operating vehicle GPS data," https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_00403602, 2018.

Feng Shan received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2015. He was a visiting student with the School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO, USA, from 2010 to 2012. He is currently an Associate Professor with the School of Computer Science and Engineering, Southeast University. His research interests include the areas of Internet of Things, wireless networks, swarm intelligence, and algorithm design and analysis.

Jianping Huang received the BS degree from Nanjing University of Science and Technology, China in 2019, and the MS degree in computer science from Southeast University, China in 2022. She is currently working in Huawei Technologies Co. Ltd. Her research interests are in energy consumption of UAV, UAV scheduling and flight planning.

Runqun Xiong received the PhD degree in computer science from Southeast University. He was with the European Organization for Nuclear Research as a Research Associate for the AMS-02 experiment from 2011 to 2012. He is currently an associate professor with the School of Computer Science and Engineering, Southeast University, China, where he is involved in AMS-02 data processing at the AMS Science Operations Center. His current research interests include cloud computing, industrial Internet, and drone-based wireless communication systems. He is a member of the ACM and the China Computer Federation.

Fang Dong is currently a professor in School of Computer Science and Engineering, Southeast University, China. He received his B.S. and M.S. degrees in Computer Science from Nanjing University of Science & Technology, China in 2004 and 2006, respectively, and received his Ph.D. degree in Computer Science from Southeast University in 2011. His current research interests include edge intelligence, cloud computing and Industrial Internet. He is a member of both IEEE and ACM, he also served as the co-chair of ACM Nanjing Chapter and the general secretary of ACM SIGCOMM China.

Junzhou Luo (Member, IEEE) received the B.Sc. degree in applied mathematics and the M.S. 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.

Suyang Wang received the BS degree in computer science from Nanjing University of Posts and Telecommunications, China, in 2011. He is currently pursuing the Ph.D. degree in computer science and engineering at Southeast University, China. He is currently working in Jiangsu Jinheng Information Technology Co., Ltd. His research interests are in the areas of cloud computing, Internet of Things, mobile edge computing, algorithm design and analys.