Low-Complexity UAV Path Planning for Distributed Data Collection

Dhanushka Kudathanthirige*, Iain B. Collings*, Stephen V. Hanly*, Hazer Inaltekin*, David Gesbert†
*School of Engineering, Macquarie University, Sydney, Australia. email: firstname.lastname@mq.edu.au
†Communication Systems Department, EURECOM, Sophia Antipolis, France. email: david.gesbert@eurecom.fr

***Abstract*—This paper presents an energy-efficient flight path planning scheme for rechargeable uncrewed aerial vehicles (UAVs) that sequentially harvest data from distributed ground nodes (GNs). The UAV has a limited battery and must return to a charging station to recharge when needed. We propose a novel low-complexity path planning method that iteratively updates the visiting sequence of GNs and their corresponding hovering locations. The path always ends at a charging station, and a new GN is added only if the charging station can still be reached. Our analysis shows that the proposed algorithm has a quadratic computational complexity with respect to the number of GNs, compared to the exponential complexity of the optimal design. Numerical results demonstrate that our scheme achieves near-optimal performance in terms of energy consumption.**

***Index Terms*—Data Harvesting, Rechargeable UAV, IoT**

I. Introduction

Uncrewed aerial vehicles (UAVs) have significant potential as aerial access points for data collection and distribution in large-scale Internet of Things (IoT) networks, such as agricultural monitoring, remote site surveillance, and urban sensing [1], [2]. In these scenarios, the UAV's flight path must efficiently cover all GNs while also accounting for its limited onboard battery capacity [1]–[3], which may require periodic visits to a charging station, adding significant challenges to the flight path design.

In this paper, we propose a novel flight path design algorithm for rechargeable UAVs that is of quadratic complexity in the number of GNs visited. The algorithm outputs a sequence of flight loops that together service all the GNs, and each flight loop is guaranteed to meet the UAV battery constraint. Each loop involves the UAV servicing a subset of the GNs that can be accommodated within a single charge. At the conclusion of a loop, the UAV is back at the charging station where it is recharged. The UAV visits the GNs and harvests their data sequentially, and during the communication, the UAV hovers at a suitable location near to the GN. The algorithm attempts to minimize the total energy consumption of the UAV, which includes the energy for flying between the hovering locations and the hovering energy required for communication. The algorithm selects the hovering location for the UAV to use when it communicates with a GN. This involves trading off flying time with hovering time, both of which consume energy.

This work was supported by the Australian Research Council's Discovery Project Funding Scheme (Project number DP200101627) and MQ Research Acceleration Grant.

Incorporating recharging stations greatly influences UAV flight path design by allowing data collection over multiple charging cycles [3]–[6]. This is specifically beneficial in distributed data harvesting applications to maximise the coverage. In contrast, UAVs without recharging support are limited by their battery capacity, which makes covering large areas challenging and requires deploying a strategically positioned fleet of UAVs to ensure complete coverage [7].

In distributed data harvesting applications, minimising the UAV's energy consumption is critical to maximising the number of GNs it can serve on a single battery charge [3], [8], [9]. Our previous work [3] introduced a dynamic programming-based framework that generates the optimal UAV flight path by jointly optimising the service locations for each GN, the visiting sequence, and the scheduling of recharging visits at the docking station. This optimal approach has exponential complexity with the number of GNs, as the visiting order optimisation resembles a multiple travelling salesman problem (TSP), which is NP-hard. To address this, an approximate visiting order can be used, as in [2]. Another alternative is to assume that all GNs within the UAV's coverage area can communicate simultaneously, removing the need to optimise the serving order [8], [9].

In this paper, we propose a low-complexity flight path design scheme to minimise the total energy consumption of rechargeable UAVs. In the proposed scheme, we construct the visiting sequence and hovering locations iteratively. It begins with a simple loop to serve a single GN, beginning at the docking station, flying to a hovering location to serve the GN, and then returning to the docking station. The loop is then extended to include additional GNs if the remaining battery allows. At each iteration, the next GN and its position in the visiting order are selected to minimise the additional travel distance added to the existing loop. For each newly added GN, the UAV hovering location is then optimised to minimise the combined energy cost of flying to that location, serving the GN, and reconnecting to the existing path. This process continues until either all GNs have been served or the remaining battery is insufficient to accommodate another GN. When the battery is insufficient, we initiate a new loop and repeat the process in the same manner until all GNs are covered.

Our scheme considers both the energy cost of flying between hovering locations and the energy required for communication while hovering. It is demonstrated through numerical results that it finds a near-optimal GN visiting sequence and hovering

locations while accounting for battery constraints. Moreover, our analysis shows that the proposed scheme has a computational complexity of quadratic order, in contrast to the exponential complexity of the optimal approach in [3].

II. System and Energy Consumption Models

We consider a UAV-assisted communication system consisting of a rotary-wing UAV that collects data from K GNs positioned at $\boldsymbol{v}_k = (v_k^{(1)}, v_k^{(2)}, 0)^T \in \mathbb{R}^3$ for $k \in [1:K]$. Each GN could be an IoT device that gathers Q_k bits of information from its environment and is required to transmit it to the UAV when it hovers nearby.

The UAV's flight path begins and ends at the docking station located at $\boldsymbol{x}_{\mathrm{D}} = (x_{\mathrm{D}}^{(1)}, x_{\mathrm{D}}^{(2)}, x_{\mathrm{D}}^{(3)})^T \in \mathbb{R}^3$, where $0 \leq x_{\mathrm{D}}^{(3)} < H$, and H is the flying height of the UAV when it flies between GNs. UAV battery recharging is also performed at this station. At the start, the UAV ascends vertically to $\boldsymbol{x}_0 \in \mathcal{P}^H$, which is $(H - x_{\mathrm{D}}^{(3)})$ m directly above the docking station.

It then flies between the GNs, hovering near each one to collect data before proceeding to the next GN, following a fly-hover-communicate protocol, such that its flight path lies on $\mathcal{P}^H = \{\boldsymbol{x} \in \mathbb{R}^3 : x^{(3)} = H\}$ plane [2], [6]. When its battery is depleted or after serving all GNs, the UAV returns to $\boldsymbol{x}_0$ and then descends vertically down to $\boldsymbol{x}_{\mathrm{D}}$ for recharging or to complete its mission. We optimise the UAV's flight path on the $\mathcal{P}^H$ plane by determining the hovering locations for each GN, the order to visit them, and the timing of returns to the docking station for recharging, such that total UAV energy consumption is minimised.

The UAV has a battery capacity of $\mathcal{E}_{\mathrm{cap}}$ Joules. It consumes δ_{up} Joules to ascend to altitude H and δ_{dn} Joules to descend back to the docking station.

We adopt the general aerial-ground wireless channel model described in [10]. Based on this model, the average time for the UAV to completely collect Q bits of data from a GN located at $\boldsymbol{v}$ while hovering at $\boldsymbol{x}$ is [2]:

$$T^{\mathrm{hov}}\left(\|\boldsymbol{v}-\boldsymbol{x}\|; Q\right) = Q\mathbb{E}\left[1/\mathcal{R}\left(\|\boldsymbol{v}-\boldsymbol{x}\|\right)\right] \tag{1}$$

where $\mathcal{R}(\|\boldsymbol{v}-\boldsymbol{x}\|)$ is the data rate in bits/s. The expectation in (1) accounts for the randomness in $\mathcal{R}$ due to the probabilistic occurrence of line-of-sight (LoS) and non-LoS conditions between the GN and the UAV. Intuitively, hovering time increases with the distance $\|\boldsymbol{v}_k - \boldsymbol{x}\|$ because both the LoS probability and the rate $\mathcal{R}$ decrease as the UAV moves further away from the GN, leading to a longer hovering time to collect Q bits. For brevity, we denote the hovering time required to serve GN k at $\boldsymbol{v}_k$ while the UAV is hovering at $\boldsymbol{x}$ as $T_k^{\mathrm{hov}}(\boldsymbol{x}) = T^{\mathrm{hov}}\left(\|\boldsymbol{v}_k - \boldsymbol{x}\|; Q_k\right)$.

The power consumption of a rotary-wing UAV flying at a constant altitude is expressed as a function of its speed V, denoted by $P_{\mathrm{UAV}}(V)$ [6]. The UAV consumes $P_{\mathrm{UAV}}(0)$ when hovering to serve a GN. According to (1), the expected energy required to hover at $\boldsymbol{x}$ and collect Q_k bits from GN k at $\boldsymbol{v}_k$ is calculated as $T_k^{\mathrm{hov}}(\boldsymbol{x})\, P_{\mathrm{UAV}}(0)$. The UAV flies at an optimal speed that minimises its energy expenditure per unit distance, and the resulting minimum energy consumption per unit distance is denoted by $E^\star$ (in J/m) [6].

III. UAV flight path Design problem

In this section, we formulate the UAV flight path design problem to minimise the total energy consumption of the UAV. Due to the UAV's battery capacity, it may return to the docking station multiple times for recharging. As a result, the flying path optimisation problem becomes a loop optimisation problem. First, we define a loop as follows.

Definition 1. *A loop $\boldsymbol{\phi}$ is defined as an ordered tuple $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{|\boldsymbol{\phi}|})$ for $\phi_i \in [1:K]$ with distinct elements, i.e., $\phi_i \neq \phi_j$ for $i, j \in \{1, \ldots, |\boldsymbol{\phi}|\}$.*

The entries of $\boldsymbol{\phi}$ specify the subset of GN indices to be served and their visiting order within the loop. We also define a function to extract the GNs included in a loop, which facilitates performing set operations on the GNs across different loops.

Let Ω denote the collection of all possible loops, and let $\mathcal{F}$ represent the set of all subsets of $[1:K]$. We define a set function $\lambda : \Omega \to \mathcal{F}$ as follows:

$$\lambda(\boldsymbol{\phi}) \triangleq \left\{\phi_1, \ldots, \phi_{|\boldsymbol{\phi}|}\right\} \tag{2}$$

For a loop $\boldsymbol{\phi}$, we define the UAV flight path as follows.

Definition 2. *UAV flight path for a loop $\boldsymbol{\phi}$ is defined as an ordered location sequence $\mathcal{T} = \left(\boldsymbol{x}_0, , \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{|\boldsymbol{\phi}|}, \boldsymbol{x}_{|\boldsymbol{\phi}|+1}\right)$ that satisfies $\boldsymbol{x}_{|\boldsymbol{\phi}|+1} = \boldsymbol{x}_0$ and $\boldsymbol{x}_i \in \mathcal{P}^H$ for $i \in [0 : |\boldsymbol{\phi}| + 1]$.*

According to the definition 2, the UAV flying path begins at $\boldsymbol{x}_0$ on the $\mathcal{P}^H$ plane. Then, it traverses through hovering locations $\boldsymbol{x}_i,\ i \in [1 : |\boldsymbol{\phi}|]$ on the same plane while serving GNs in the order defined in $\boldsymbol{\phi}$. In particular, the UAV serves GN $\boldsymbol{\phi}(i)$ while hovering at $\mathcal{T}(i) = \boldsymbol{x}_i$. After serving the last GN in the loop $\phi_{|\boldsymbol{\phi}|}$ while hovering at $\boldsymbol{x}_{|\boldsymbol{\phi}|}$, UAV returns to $\boldsymbol{x}_0$. The initial ascent and final descent actions are excluded from the definition 2, as these manoeuvres are fixed and known. However, their associated energy consumption is included in the calculations when determining the loops' and total energy consumption, as discussed below.

The energy consumption for serving GNs in loop $\boldsymbol{\phi}$, along the path $\mathcal{T}$, hovering at locations $\{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}$ is denoted as $J_{\mathrm{loop}}\left(\boldsymbol{\phi}, \{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}; \boldsymbol{x}_0\right)$, and can be expressed as in (3). In (3), the first sum term computes the total hovering energy in the loop, which depends on the chosen service locations, $\{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}$, and their associated GNs $\boldsymbol{\phi}$. The second sum term accounts for the energy consumed in travelling between hovering locations within the loop. The third term captures the energy required to travel from the loop's starting point $\boldsymbol{x}_0$ (after recharging and ascent) to the first service location $\boldsymbol{x}_1$, as well as the energy required to travel from the last service location $\boldsymbol{x}_{|\boldsymbol{\phi}|}$ back to $\boldsymbol{x}_0$ at the end of the loop, either for recharging or mission completion. The final term, $\delta_{\mathrm{up}} + \delta_{\mathrm{dn}}$, accounts for the energy consumed by the UAV when ascending from the docking station $\boldsymbol{x}_{\mathrm{D}}$ to $\boldsymbol{x}_0$ at the start of the loop and descending back to $\boldsymbol{x}_{\mathrm{D}}$ at the end of the loop.

$$J_{\text{loop}}\left(\boldsymbol{\phi}, \{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}; \boldsymbol{x}_0\right) = P_{\text{UAV}}(0) \sum_{k=1}^{|\boldsymbol{\phi}|} T_{\phi_k}^{\text{hov}}(\boldsymbol{x}_k) + \sum_{k=1}^{|\boldsymbol{\phi}|-1} E^* \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| + E^* \left(\|\boldsymbol{x}_1 - \boldsymbol{x}_0\| + \left\|\boldsymbol{x}_0 - \boldsymbol{x}_{|\boldsymbol{\phi}|}\right\|\right) + \delta_{\text{up}} + \delta_{\text{dn}} \quad (3)$$

A loop $\boldsymbol{\phi}$ and its service locations $\{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}$ are feasible if and only if that loop can be covered within the battery capacity, i.e., $J_{\text{loop}}(\boldsymbol{\phi}, \{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}; \boldsymbol{x}_0) \leq \mathcal{E}_{\text{cap}}$. Recall that the UAV's battery is charged to $\mathcal{E}_{\text{cap}}$ at the docking station, which is the maximum possible energy that can be spent in the entire loop, ensuring safe return of the UAV.

A loop-set $\Phi \subseteq \Omega$ is defined as a collection of loops with the flying paths $\{\mathcal{T}_l\}_{l=1}^{|\Phi|}$ that together serve all GNs. The total energy consumption of the UAV to complete all loops in $\Phi = \{\boldsymbol{\phi}_l\}_{l=1}^{|\Phi|}$ can be computed as the sum of each loop's energy consumption

$$J_{\text{Total}}\left(\Phi, \{\boldsymbol{x}_{l,k}\}_{\substack{l\in[1:|\Phi|] \\ k\in[1:|\boldsymbol{\phi}_l|]}}; \boldsymbol{x}_0\right) = \sum_{l=1}^{|\Phi|} J_{\text{loop}}\left(\boldsymbol{\phi}, \{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}_l|}; \boldsymbol{x}_0\right)$$

subject to the battery constraint. Hence, the total UAV energy consumption minimisation problem can be formulated as

$$\begin{aligned} \underset{\Phi\subseteq\Omega,\ \boldsymbol{x}_{l,k}\in\mathbb{R}^3}{\text{minimize}} \quad & J_{\text{Total}}\left(\Phi, \{\boldsymbol{x}_{l,k}\}_{\substack{l\in[1:|\Phi|] \\ k\in[1:|\boldsymbol{\phi}_l|]}}; \boldsymbol{x}_0\right) \\ \text{subject to} \quad & J_{\text{loop}}(\boldsymbol{\phi}_l, \{\boldsymbol{x}_{l,k}\}_{k=1}^{|\boldsymbol{\phi}_l|}; \boldsymbol{x}_0) \leq \mathcal{E}_{\text{cap}}, \forall l \in [1:|\Phi|] \\ & x_{l,k}^{(3)} = H, \forall k \in [1:|\boldsymbol{\phi}_l|], l \in [1:|\Phi|] \\ & \bigcup_{l=1}^{|\Phi|} \lambda(\boldsymbol{\phi}_l) = [1:K], \\ & \lambda(\boldsymbol{\phi}_l) \bigcap \lambda(\boldsymbol{\phi}_{l'}) = \emptyset, \text{ for } l \neq l', l, l' \in [1:|\Phi|] \end{aligned} \quad (4)$$

The first constraint in (4) imposes the battery capacity limit on the energy consumption on each loop. The second constraint enforces that the UAV maintains a constant altitude H throughout its flight, meaning its path is optimised on the $\mathcal{P}^H$ plane. The third and fourth constraints in (4) relate to partitioning the GN set among the loops: the third constraint ensures that all GNs are visited by the end of the mission, while the fourth constraint guarantees that each GN is visited at most once.

Note the overall problem (4) is feasible if each node is within range of the charging station such that the UAV could reach it (and hover long enough to serve it) with a full battery charge (i.e. on a direct out-and-back flight). This is because the set of feasible trajectories includes the possibility of returning to the charging station between each node. This condition can be easily checked before attempting to solve (4).

The optimal flight path design for the problem formulated in (4) was presented in [3] using a dynamic programming-based framework. This approach jointly optimised the hovering locations (selected from a pre-defined grid), the visiting sequence of these locations, and the timing for returning to the recharging station. Nevertheless, the complexity of the algorithm grows exponentially with the number of GNs, K. This exponential complexity arises from the need to determine the optimal visiting order, which corresponds to the classical TSP. Motivated by this, in this paper, we explore a low-complexity flight path design that can efficiently minimise the UAV's total energy consumption.

IV. Proposed Low Complexity Algorithm

In this section, we present our heuristic-based low-complexity algorithm. To achieve this, we decompose the original flight path design problem in (4) into two sub-problems: (i) determining the hovering locations and (ii) optimising the loop set. These sub-problems are solved iteratively until all GNs are assigned to loops, while minimising the UAV's energy consumption, and satisfying the battery constraint.

A. Hovering Location Calculation

In this subsection, we describe the procedure for determining the hovering location of a GN newly assigned to an existing loop $\boldsymbol{\phi}$ with flight path $\mathcal{T}$. Suppose the new GN is located at $\boldsymbol{v}$ (where the index k is omitted for brevity) and is to be served by introducing a new hovering location $\boldsymbol{x}$ between $\mathcal{T}(i) = \boldsymbol{x}_i$ and $\mathcal{T}(i+1) = \boldsymbol{x}_{i+1}$ for $i \in [0:|\boldsymbol{\phi}|]$. In this case, the UAV performs the following sequence of actions: it travels from $\boldsymbol{x}_i$ to $\boldsymbol{x}$, hovers at $\boldsymbol{x}$ to serve a new GN, and travels to $\boldsymbol{x}_{i+1}$. The energy cost of this action is computed as

$$\begin{aligned} J(\boldsymbol{x}; \boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \boldsymbol{v}) = & E^\star (\|\boldsymbol{x} - \boldsymbol{x}_i\| + \|\boldsymbol{x}_{i+1} - \boldsymbol{x}\|) \\ & + P_{\text{UAV}}(0) T^{\text{hov}}(\boldsymbol{x}) \end{aligned} \quad (5)$$

where the first term denotes the UAV's total travel distance, while the second represents the hovering energy at $\boldsymbol{x}$. Our goal is to find the optimal service location $\boldsymbol{x}^*$ that minimises energy consumption for this action.

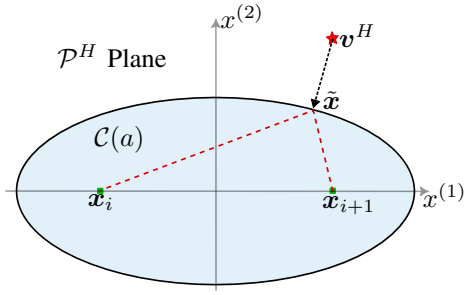


Fig. 1: A UAV flies $\boldsymbol{x}_i$ to $\boldsymbol{x}$, hovers at $\boldsymbol{x}$, proceeds to $\boldsymbol{x}_{i+1}$.

The process of serving a new GN can be formulated as a geometric projection problem (see Fig.1). Without loss of generality, we set $\boldsymbol{x}_i = (-c, 0, H)^T$, $\boldsymbol{x}_{i+1} = (c, 0, H)^T$, and projected the GN's location $\boldsymbol{v}$ onto the $\mathcal{P}^H$ plane as $\boldsymbol{v}^H = (v^{(1)}, v^{(2)}, H)^T$. In Fig.1, the travel distance $a \in \mathbb{R}_+$ forms an ellipse with foci at $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$. The convex set $\mathcal{C}(a) = \{\|\boldsymbol{x} - \boldsymbol{x}_i\| + \|\boldsymbol{x}_{i+1} - \boldsymbol{x}\| \leq a\}$ represents all possible service locations on the $\mathcal{P}^H$ plane that the UAV can reach with a maximum travel distance of a.

The travel distance is constrained as $a \in [a_{\min}, a_{\max}]$. Here, $a_{\min} = \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|$ corresponds to the UAV flying directly from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{i+1}$ while serving the GN at its closest point along

this straight path. The scenario where the UAV hovers directly above the GN corresponds to $a_{\max} = \|\boldsymbol{x} - \boldsymbol{x}_i\| + \|\boldsymbol{x}_{i+1} - \boldsymbol{x}\|$. For a given a, the optimal hovering location that minimises the energy consumption, $\boldsymbol{x}^*(a)$, is given by the following lemma.

Lemma 1. *Given UAV travel distance $a \in [a_{\min}, a_{\max}]$ for the scenario illustrated in Fig. 1, the projection of $\boldsymbol{v}^H \in \mathcal{P}^H$ onto the convex set $\mathcal{C}(a)$, denoted by $\boldsymbol{x}^*(a) = \mathrm{P}_{\mathcal{C}(a)}\left(\boldsymbol{v}^H\right)$, minimises the UAV energy consumption.*

Proof. This follows since $\boldsymbol{x}^*(a)$ minimises the hovering distance to $\boldsymbol{v}$ by definition. ∎

This lemma is important because, given UAV travel distance, we can compute $\boldsymbol{x}^*(a)$ using standard convex optimisation algorithms. In addition, utilising Lemma 1, the optimal hovering location for serving GN at $\boldsymbol{v}$ can be computed via one-dimensional search for optimum a (denoted by $a^\star$) by solving

$$a^\star = \underset{a \in [a_{\min}, a_{\max}]}{\operatorname{argmin}} \; E^\star a + P_{\mathrm{UAV}}(0)\, T_{\mathrm{hov}}\left(\mathrm{P}_{\mathcal{C}(a)}\left(\boldsymbol{v}^H\right)\right) \quad (6)$$

and computing the projection

$$\boldsymbol{x}^\star(a^\star) = \mathrm{P}_{\mathcal{C}(a^\star)}\left(\boldsymbol{v}^H\right). \quad (7)$$

Note that the first term in (6) increases linearly with a and the second term decreases monotonically with a. Therefore, the optimal solution a^* should balance the energy consumption for travel and hovering.

B. Updating Loops

The UAV flight path $\mathcal{T}$ within a loop $\boldsymbol{\phi}$ consists of a sequence of waypoints which, when connected in the order specified by $\mathcal{T}$, define the loop's flight path. Each waypoint corresponds to a GN indicated by $\boldsymbol{\phi}$. To admit the next GN into the loop ϕ, its index must be inserted into $\boldsymbol{\phi}$, and its corresponding hovering location added to $\mathcal{T}$. We choose to do this such that the next GN to be added to an existing loop will minimise the additional length it adds to the current loop trajectory. Obviously, this addition is only made if the energy consumption of the updated flight path remains within the battery capacity $\mathcal{E}_{\mathrm{cap}}$; otherwise, a new loop is initiated. Also, for each GN admitted, it is necessary to check whether the new line segments intersect with the existing lines in the trajectory, and if so, the loop $\boldsymbol{\phi}$ and the waypoints in T must be reordered to eliminate any crossings. The process of initiating a loop, updating an existing loop, and the crossing detection process are summarised in algorithm 1.

1) Initiating a loop: We denote the set of GNs already assigned to a loop as $\mathcal{U}$, and the set of GNs yet to be assigned as $\overline{\mathcal{U}} = [1:K] \setminus \mathcal{U}$. Initially, $\mathcal{U} = \emptyset$ and $\overline{\mathcal{U}} = [1:K]$.

To initiate a new loop, first, the travel distance required to serve each GN in $\overline{\mathcal{U}}$ individually is calculated. The flight path for serving a single GN is constructed as follows: the UAV departs from $\boldsymbol{x}_0$, travels along the straight line towards $\boldsymbol{v}_k^H$ (the projection of $\boldsymbol{v}_k$ onto the $\mathcal{P}^H$), hovers at an optimally chosen point along this path to harvest data, and then returns to $\boldsymbol{x}_0$. The hovering location is determined to minimise the UAV's energy consumption. This heuristic selection strategy prioritises the most distant hovering location first, and then extends the loop by adding other GNs along similar directions until the total energy consumption of the loop reaches the UAV's battery capacity. Compute the energy-minimising hovering distance from $\boldsymbol{x}_0$ for individually serving GN k as

$$d_k^\star = \underset{0 \le d \le \left\|\boldsymbol{v}_k^H - \boldsymbol{x}_0\right\|}{\operatorname{argmin}} \; 2E^\star d + P_{\mathrm{UAV}}(0) T_k^{\mathrm{hov}}\left(\boldsymbol{x}_0 + d\boldsymbol{u}_k\right) \quad (8)$$

where $k \in \overline{\mathcal{U}}$ and $\boldsymbol{u}_k$ is the unit vector given by $\boldsymbol{u}_k = \frac{\left(\boldsymbol{v}_k^H - \boldsymbol{x}_0\right)}{\left\|\boldsymbol{v}_k^H - \boldsymbol{x}_0\right\|}$. The GN selected to initiate the loop is given by

$$k_1 = \underset{k \in \overline{\mathcal{U}}}{\operatorname{argmax}}\; d_k \quad (9)$$

2) Finding next GN: Consider an existing loop $\boldsymbol{\phi}$ with a flying path $\mathcal{T} = \left(\boldsymbol{x}_0, , \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{|\boldsymbol{\phi}|}, \boldsymbol{x}_0\right)$, where $|\boldsymbol{\phi}| \le K$ and $\mathcal{U} \neq \emptyset$. The energy consumption for this flight path is given by $J_{\mathrm{t}} = J_{\mathrm{loop}}(\boldsymbol{\phi}, \{\boldsymbol{x}_k\}_{k=1}^{|\boldsymbol{\phi}|}; \boldsymbol{x}_0)$, and it satisfies $J_{\mathrm{t}} < \mathcal{E}_{\mathrm{cap}}$, indicating that there could be sufficient remaining battery to serve at least one more GN within the loop.

The index of the new GN to be added, along with its insertion point in the loop $\boldsymbol{\phi}$, is determined as follows. We first assume that the UAV can serve one of the GNs in $\overline{\mathcal{U}}$ by deviating from its original path $\mathcal{T}$ at hovering location $\boldsymbol{x}_i$ for $i \in [0 : |\boldsymbol{\phi}|]$, hovering above the new GN $\boldsymbol{x}_{\mathrm{t}} = \boldsymbol{v}_k^H \; \forall k \in \overline{\mathcal{U}}$, and then returning to $\boldsymbol{x}_{i+1}$ before continuing along the original path. This *tentative* hovering location $\boldsymbol{x}_{\mathrm{t}}$ will be updated later once the new loop is selected. If this choice of hovering location is feasible, then the updated loop is guaranteed to remain within the UAV's battery capacity.

Denote the new trajectory by $\mathcal{T}' = (\{\mathcal{T}\}_{j=0}^{i}, \boldsymbol{x}, \{\mathcal{T}\}_{j=i+1}^{|\boldsymbol{\phi}|+1})$, $i \in [0 : |\boldsymbol{\phi}|]$. We select both the GN index and its insertion point to the loop by minimising the path elongation over the remaining GNs $\overline{\mathcal{U}}$ and path segments in $\mathcal{T}$ as follows:

$$(k_{\mathrm{t}}, i_{\mathrm{t}}) = \underset{k \in \overline{\mathcal{U}}, i \in [0:|\boldsymbol{\phi}_{\mathrm{t}}|]}{\operatorname{argmin}} \; \left(\left\|\boldsymbol{x}_i - \boldsymbol{v}_k^H\right\| + \left\|\boldsymbol{v}_k^H - \boldsymbol{x}_{i+1}\right\| - \left\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\right\|\right) \quad (10)$$

Following lemma 1, the hovering location for GN k_{t} can be updated as $\boldsymbol{x}_{\mathrm{t}} \leftarrow \mathrm{P}_{\mathcal{C}(\tilde{a})}\left(\boldsymbol{v}_{k_{\mathrm{t}}}^H\right)$.

3) Crossing detection and resolution: We compute new tentative flying path and the visiting order in the loop as $\mathcal{T}_{\mathrm{t}} = (\{\mathcal{T}\}_{j=0}^{i_{\mathrm{t}}}, \boldsymbol{x}_{\mathrm{t}}, \{\mathcal{T}\}_{j=i_{\mathrm{t}}+1}^{|\boldsymbol{\phi}|+1})$ and $\boldsymbol{\phi}_t \leftarrow \left(\{\boldsymbol{\phi}\}_1^i, k, \{\boldsymbol{\phi}\}_{i+1}^{|\boldsymbol{\phi}|}\right)$. The path $\mathcal{T}_{\mathrm{t}}$ may contain crossings between line segments due to the newly added segments $(\boldsymbol{x}_{i_{\mathrm{t}}}, \boldsymbol{x}_{\mathrm{t}})$ and $(\boldsymbol{x}_{\mathrm{t}}, \boldsymbol{x}_{i_{\mathrm{t}}+1})$ intersecting with segments in the original path $\mathcal{T}$. A minimal distance path that traverses $\{\boldsymbol{x}_i\}_{i=0}^{|\boldsymbol{\phi}_t|}$ cannot intersect itself [11]. Any crossing implies that the UAV is using more energy to visit the set of hovering locations than the minimal path connecting the hovering locations. It is possible to detect such intersecting segments and to reorder the visiting order to eliminate crossings and reduce energy consumption. Algorithm 2 outlines the procedure for detecting path crossings and resolving them when they occur. The operation flip() refers to reversing the order of the input sequence.

Algorithm 1 : UAV Trajectory Design Algorithm

Input: $\{Q_k\}_{k\in[1:K]}$, $\{\boldsymbol{v}_k\}_{k\in[1:K]}$, $E^\star$, $P_{\text{UAV}}(0)$, $\mathcal{E}_{\text{cap}}$, δ_{dn}, δ_{up}, and $\boldsymbol{x}_0$.
Output: Loop set $\Phi = \{\boldsymbol{\phi}_l\}_{l=1}^{|\Phi|}$ and service locations $\{\boldsymbol{x}_{l,k}\}_{l\in[1:|\Phi|],\, k\in[1:|\boldsymbol{\phi}_l|]}$
Initialise: $\tilde{J}_{\text{loop}} = 0$, $\tilde{J}_{\text{Total}} = 0$, $l = 1$ ▷ Loop index
$\overline{\mathcal{U}} = [1:K]$ ▷ Set all GNs as remaining GNs
Compute $\{d_k\}_{k\in\overline{\mathcal{U}}}$ according to (8).
while $\overline{\mathcal{U}} \neq \emptyset$ **do**
  if $\tilde{J}_{\text{loop}} = 0$ **then** ▷ Start new loop
    Initialise empty $\boldsymbol{\phi}_l$ and $\mathcal{T}_l$. Compute k_1 using (9)
    $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_0 + d_{k_1}\boldsymbol{u}_{k_1}$ where $\boldsymbol{u}_i$ is defined in (8)
    $\boldsymbol{\phi}_{\text{t}} \leftarrow k_1$, $\tilde{J}_{\text{t}} \leftarrow J(\boldsymbol{x}_{\text{t}}; \boldsymbol{x}_0, \boldsymbol{x}_0, \boldsymbol{v}_{k_1})$, $\mathcal{T}_{\text{t}} \leftarrow (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_0)$ ▷ Update loop, loop cost, and flight path
  else
    $\boldsymbol{\phi}_{\text{t}} = \boldsymbol{\phi}_l$ and $\mathcal{T}_{\text{t}} = \mathcal{T}_l$,
    Find next GN k_{t} and insertion point i_{t} from (10)
    Compute $\boldsymbol{x}_{\text{t}}$ for GN k_{t} using (7)
    Run algorithm 2 to detect crossing and modify path $\left[\mathcal{T}_{alg}, \boldsymbol{\phi}_{alg}, \tilde{J}_{alg}\right] = \text{Algorithm2}(\mathcal{T}_t, \boldsymbol{\phi}, \boldsymbol{x}_t, k_{\text{t}}, i_{\text{t}})$
    $\mathcal{T}_{\text{t}} \leftarrow \mathcal{T}_{alg}$, $\boldsymbol{\phi}_{\text{t}} \leftarrow \boldsymbol{\phi}_{alg}$, $\tilde{J}_{\text{t}} \leftarrow \tilde{J}_{alg}$
  end if
  if $\tilde{J}_{\text{t}} \leq \mathcal{E}_{\text{cap}}$ **then**
    $\mathcal{T}_l \leftarrow \mathcal{T}_{\text{t}}$, $\boldsymbol{\phi}_l \leftarrow \boldsymbol{\phi}_{\text{t}}$ ▷ Update path and visiting order
    $\tilde{J}_{\text{loop}} \leftarrow \tilde{J}_{\text{t}}$, $\tilde{J}_{\text{Total}} \leftarrow \tilde{J}_{\text{Total}} + \tilde{J}_{\text{loop}}$ ▷ Update energy
    $\overline{\mathcal{U}} \leftarrow \overline{\mathcal{U}}/\lambda(\boldsymbol{\phi}_l)$ ▷ Remove served GNs
  else
    $\tilde{J}_{\text{loop}} \leftarrow 0$ ▷ Reset loop energy
    $l = l + 1$ ▷ Update the loop index
  end if
end while

Algorithm 2 : Crossing Detection and Resolution

Input: $\mathcal{T}$, $\boldsymbol{\phi}$, $\tilde{\boldsymbol{x}}$, i, k
Initialise: Setup tentative flying path and visiting order $\mathcal{T}_t \leftarrow \left(\{\mathcal{T}\}_1^i, \tilde{\boldsymbol{x}}, \{\mathcal{T}\}_{i+1}^{|\boldsymbol{\phi}|+1}\right)$, $\boldsymbol{\phi}_t \leftarrow \left(\{\boldsymbol{\phi}\}_1^i, k, \{\boldsymbol{\phi}\}_{i+1}^{|\boldsymbol{\phi}|}\right)$
for new edges $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ in $\mathcal{T}_t$ **do**
  for $j = 0 : |\mathcal{T}_t| - 1$, $j \neq i$ **do** ▷ All edges in $\mathcal{T}_t$
    Check for crossing of $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ and $(\boldsymbol{x}_j, \boldsymbol{x}_{j+1})$ using Algorithm 3
    if Crossing happens **then**
      $i_a = \min(i, j)$ and $i_b = \max(i, j)$
      $\mathcal{T}_t \leftarrow \left(\{\mathcal{T}\}_1^{i_a}, \text{flip}\left(\{\mathcal{T}\}_{i_a+1}^{i_b}\right), \{\mathcal{T}\}_{i_b+1}^{|\boldsymbol{\phi}|_t+1}\right)$
      $\boldsymbol{\phi}_t \leftarrow \left(\{\boldsymbol{\phi}\}_1^{i_a}, \text{flip}\left(\{\boldsymbol{\phi}\}_{i_a+1}^{i_b}\right), \{\boldsymbol{\phi}\}_{i_b+1}^{|\boldsymbol{\phi}|_t}\right)$
    end if
  end for
end for
$\tilde{J}_t = J_{\text{loop}}(\phi_t, \{\boldsymbol{x}_i\}; \boldsymbol{x}_0)$ ▷ Energy cost of new loop $\boldsymbol{\phi}_t$
$\mathcal{T}_{nw} \leftarrow \mathcal{T}_t$, $\boldsymbol{\phi}_{nw} \leftarrow \boldsymbol{\phi}_t$, $\tilde{J}_{nw} \leftarrow \tilde{J}_t$
Return: $\mathcal{T}_{nw}$, $\boldsymbol{\phi}_{nw}$, $\tilde{J}_{nw}$

Algorithm 3 :Line Segment Crossing Detection

Input: Two line segments $(\boldsymbol{y}_1, \boldsymbol{z}_1)$ and $(\boldsymbol{y}_2, \boldsymbol{z}_2)$
Compute $O_1 = \text{OT}(\boldsymbol{y}_1, \boldsymbol{z}_1, \boldsymbol{y}_2)$, $O_2 = \text{OT}(\boldsymbol{y}_1, \boldsymbol{z}_1, \boldsymbol{z}_2)$, $O_3 = \text{OT}(\boldsymbol{y}_2, \boldsymbol{z}_2, \boldsymbol{y}_1)$, and $O_1 = \text{OT}(\boldsymbol{y}_2, \boldsymbol{z}_2, \boldsymbol{z}_1)$
  XFlag = False
if $O_1 \neq O_2$ & $O_3 \neq O_4$ **then**
  XFlag = True
end if
Return: XFlag

The crossings of two line segments can be detected using an *orientation test*(OT) using coordinates of the segments [12, Ch. 33]. Algorithm 3 summarises the procedure for detecting whether two line segments, $(\boldsymbol{y}_1, \boldsymbol{z}_1)$ and $(\boldsymbol{y}_2, \boldsymbol{z}_2)$, intersect. The algorithm outputs a Boolean value, returning 'True' if the segments intersect and 'False' otherwise.

V. Numerical Results

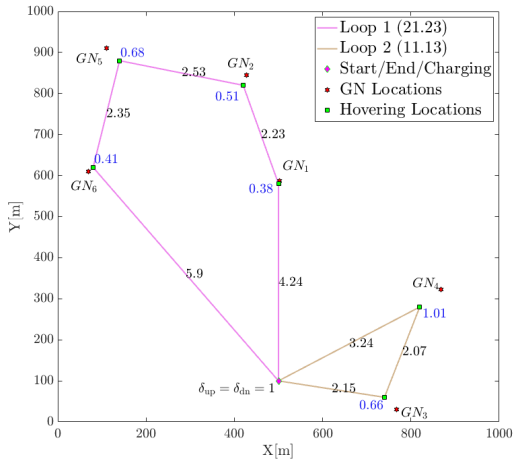
The UAV operates at an altitude of $H = 100\,\text{m}$ to serve GNs distributed over a square area of 1 sqkm. The docking station is at $\boldsymbol{x}_D = (500, 100, 0)^T\,\text{m}$. The probabilistic LoS/NLoS modelling from [10] is used, with a path-loss exponent of 2.7 and the bandwidth 50 MHz. UAV's power consumption model in [3] is adopted. The UAV battery capacity is $\mathcal{E}_{\text{cap}} = 22\,\text{kJ}$.

We compare our low-complexity trajectory design with the following two baseline approaches:

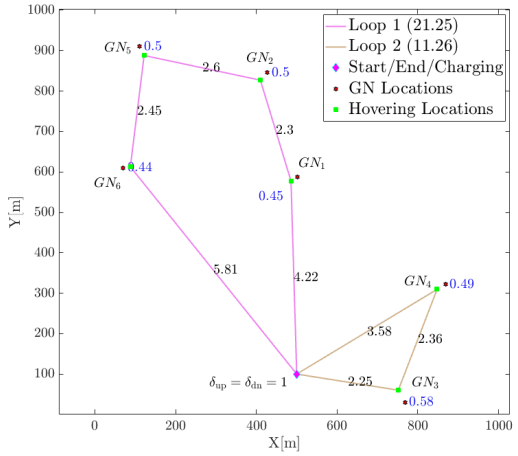
- *Full-DP scheme:* This scheme designs the full flight path of a UAV with a limited battery by jointly determining the GN visiting order, hovering locations, and the best times for recharging, as described in Algorithm 1 of [3]. The hovering locations are chosen from a predefined grid over $\mathcal{P}^H$ with $MN = 50 \times 50$ grid points. The battery is also quantised with a resolution of $0.25\,\text{kJ}$ ($B = 89$ levels).
- *2 Approx-DP scheme* This scheme is inspired by Algorithms 2 and 3 from [2], which decompose the trajectory design problem into two sub-problems: determining the visiting order and selecting the hovering locations. The visiting order is approximated using a 2-approximation solution to the TSP for the GNs [2, Algo. 3], and the hovering locations are then optimised using a DP approach [2, Algo. 2] based on this visiting order. We modified the algorithms to enforce the battery constraint, directing the UAV to return to the recharging station when needed, while otherwise following the visiting order from [2, Algo. 3] and hovering locations from [2, Algo. 2].

The complexity of the proposed algorithm 1 is in the order of $\mathcal{O}(K^2)$, which is significantly lower compared to the Full-DP scheme, which incurs an exponential complexity of $\mathcal{O}((MNB)^2K2^K)$. It also improves upon the 2-Approx DP scheme, which has a complexity of $\mathcal{O}((MN)^2K + K^2\log K)$.

Fig. 2 presents the paths generated by the Full-DP scheme and the proposed method for a generic scenario with six GNs, each required to transmit 500 kbits. In the figures, the energy costs for travelling between hovering locations are annotated along each line segment, while the hovering energy costs for



(a) UAV flight path for Full-DP scheme (optimal).



(b) UAV flight path for the proposed algorithm.

Fig. 2: Flight path comparison with Full-DP for $\mathcal{E}_{\text{cap}} = 22\,\text{kJ}$.

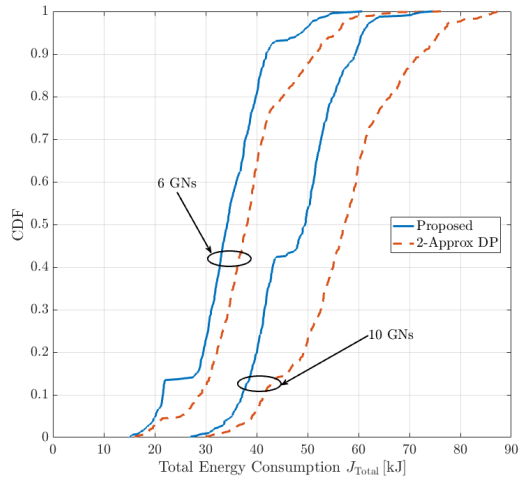


Fig. 3: Total energy comparison: proposed vs. 2-approx DP.

data collection from each GN are indicated in blue next to their corresponding hovering locations.

Fig. 2(a) shows the optimal UAV flight path produced by the Full-DP scheme, which serves six GNs across two loops, resulting in a total energy consumption of 32.36 kJ. For comparison, Fig. 2(b) illustrates the flight path generated by our proposed scheme using Algorithm 1. Notably, our scheme produces the same loop structure as the Full-DP scheme in this scenario, with a total energy consumption of 32.51 kJ, which is only marginally higher than that of the Full-DP scheme.

Fig. 3 shows the cumulative distribution of total energy consumption for scenarios with 6 and 10 GNs, each having 500 kbits to transmit. The results are obtained by randomly placing GNs within a 1 sqkm area and designing the UAV's flight path using both the proposed scheme and the 2-approx DP baseline. As shown, the proposed scheme consistently outperforms the baseline. This improvement arises because the proposed scheme resolves flight path crossings within each loop, thereby reducing the UAV's overall travel energy.

VI. Conclusion

We proposed an energy-efficient flight path design for a rechargeable UAV tasked with sequentially collecting data from distributed GNs. We developed a novel low-complexity heuristic-based path planning algorithm that iteratively refines both the visiting sequence of ground nodes and their corresponding hovering locations. The proposed scheme has quadratic computational complexity in the number of GNs, offering a significant improvement over the exponential complexity of the optimal design.

References

[1] Y. Wang, Z. Su, N. Zhang, and R. Li, "Mobile wireless rechargeable UAV networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 60, no. 3, pp. 33–39, 2022.

[2] D. Kudathanthirige, H. Inaltekin, S. V. Hanly, and I. B. Collings, "Optimum UAV trajectory design for data harvesting from distributed nodes," *IEEE Trans. Commun.*, vol. 72, no. 1, pp. 302–316, 2024.

[3] ——, "Optimum path planning for large scale distributed IoT data collection using a rechargeable UAV," *IEEE Open J. Commun. Soc.*, vol. 6, pp. 5155–5172, 2025.

[4] R. Wang, D. Li, Q. Wu, K. Meng, B. Feng, and L. Cong, "Rechargeable UAV trajectory optimization for real-time persistent data collection of large-scale sensor networks," *IEEE Trans. Commun.*, pp. 1–1, 2024.

[5] C. Zhan and H. Lai, "Energy minimization in Internet-of-Things system based on rotary-wing UAV," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1341–1344, 2019.

[6] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, 2019.

[7] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5g and beyond: Recent advances and future trends," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2019.

[8] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 53 172–53 184, 2020.

[9] K. Zhang, J. Cao, L. Wang, and Y. Zhang, "Green offloading and trajectory scheduling of rechargeable UAVs in aerial edge networks," in *Proc. 2022 IEEE Global Commun. Conf. (GLOBECOM 2022)*, 2022, pp. 1752–1757.

[10] I. Mohammed, I. B. Collings, and S. V. Hanly, "A new connectivity model for UAV communications and flying height optimization," *Trans. on Emerging Telecom. Technologies*, vol. 34, no. 6, pp. 1–20, Jun. 2023.

[11] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, pp. 61–75, 1956.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.