Practical Mission Planning for Optimized UAV-Sensor Wireless Recharging

Qiuchen Qian, James O’Keeffe, Yanran Wang and David Boyle

Abstract—Optimal maintenance of sensor nodes in a Wireless Rechargeable Sensor Network (WRSN) requires effective scheduling of power delivery vehicles by solving the Charging Scheduling Problem (CSP). Deploying Unmanned Aerial Vehicles (UAVs) as mobile chargers has emerged as a promising solution due to their mobility and flexibility. The CSP can be formulated as a Mixed-Integer Non-Linear Programming problem whose optimization objective is maximizing the recharged energy of sensor nodes within the UAV battery constraint. While many studies have demonstrated satisfactory performance of heuristic algorithms in addressing specific routing problems, few studies explore online updating (i.e., mission re-planning ’on the fly’) in the CSP context. Here we present a new offline and online mission planner leveraging a first-principles power consumption model that uses real-time state information and environmental information. The planner, namely Rapid Online Metaheuristic-based Planner (ROMP), supplements solutions from a Guided Local Search (GLS) with our Context-aware Black Hole Algorithm. Our results demonstrate that ROMP outperforms GLS in most cases tested. We developed and proposed FastROMP to speed up the online mission (re-)planning algorithm by introducing a new online adjustment operator that uses the latest state information as input, eliminating the need for re-initialization. FastROMP not only provides a better quality route, but it also significantly reduces computational time. The reduction ranges from 39.57% in sparse deployment to 93.3% in denser deployments.

Index Terms—Wireless Rechargeable Sensor Network; Metaheuristic Algorithm; UAV Mission Planning; Dynamic Travelling Salesman Problem; Dynamic Orienteering Problem; Energy-constrained Optimization; Dynamic Environment.

1 INTRODUCTION

Wireless Rechargeable Sensor Network (WRSN) systems are an emerging paradigm within Internet-of-Things (IoT) technologies [1]. Maintaining a field of wireless sensors can be challenging due to the need for human intervention and costly infrastructure support. One solution to this challenge is deploying autonomous vehicles to perform maintenance tasks, such as mobile remote charging [2]. However, efficiently servicing large-scale networks using autonomous vehicles is a complex navigational problem. This paper addresses mobile chargers’ dynamic Charging Scheduling Problem (CSP) in WRSNs. Autonomous Unmanned Aerial Vehicles (UAVs), such as multi-rotor copters, are well-suited to this task in large-scale WRSN scenarios due to their agility and flexibility [3]. UAVs can use Inductive Power Transfer (IPT) to recharge sensor nodes rapidly [4], [5]. We approach the CSP as a multi-objective optimization problem where the UAV must wirelessly recharge a specific number of sensor nodes. The goal is to maximize the recharged energy of all devices deployed across a field while minimizing the UAV’s discharged energy. We define two sub-problems from different perspectives to achieve the overall optimization objective:

- Travelling Salesman Problem (TSP): A single UAV is assigned to charge all unvisited sensors in a given WRSN. Multiple iterations may be required due to the UAV battery’s limited capacity. The optimization objective of a single iteration is to minimize the UAV’s discharged energy.
- Orienteering Problem (OP): A single UAV is assigned to charge all unvisited sensors in a given WRSN. Each sensor is associated with a biased ‘prize’ weight. This model allows a temporary drop of some low-prize sensors. The optimization objective is to maximize the collected prizes, subject to a budget constraint on UAV’s discharged energy.

The TSP and OP (see Fig. 1) are classified as NP-hard problems, which means finding an optimal solution requires an exhaustive search of all possible solutions. While exact and approximation algorithms can guarantee the solution’s optimality [6], their practical computational expense increases significantly with the increase of the instance scale. Therefore, heuristic algorithms that can quickly converge to (near-)optimal solutions are generally preferred in most computation-sensitive scenarios as they empirically show a good balance of computation time and solution quality [7]. Single-solution-based meta-heuristics, such as Simulated Annealing (SA), Tabu Search (TS) and Guided Local Search (GLS) [8], modify and improve single elements in the candidate solution. In contrast, population-based meta-heuristics, such as Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Black Hole Algorithm [9], search for potential solutions by modifying multiple elements simultaneously.

A hybrid offline and online scheme is typical for optimizing UAV path planning in a dynamic environment, e.g., obstacle avoidance [10], [11]. The offline algorithm searches for a path that shortens travel time and avoids obstacles from a static safety index map. The online algorithm quickly adjusts the offline path to avoid dynamic or unexpected obstacles from a dynamic map. Similarly, due to the dynamic nature of the environment in which UAVs operate and the presence of unforeseen adversarial factors such as turbulence or strong winds, it is impossible to solve the CSP precisely, using only static information. Therefore, an
online approach that can adapt to changing environmental conditions is necessary to complete the UAV’s mission successfully.

Moreover, an appropriate power consumption model for the UAV is crucial in evaluating its capability to complete missions under varied scenarios. Without prior knowledge of the UAV’s capabilities, the power model can effectively translate the UAV’s actions into actual energy consumption. This helps the algorithm to better formulate the fitness function. Efforts to analyze power consumption models of multi-rotor UAVs operating in dynamic environments are regularly reported, e.g., in [12], [13], [14]. Many studies use anemometers to measure wind field variations, which are commonly recognized as the primary influencing factor on the power consumption model in such environments. To record real-time wind rose and relative instantaneous power, Rodrigues et al. deployed an anemometer and a power sensor on the DJI Matrice 100 (M100) drone [14]. Their work indicates extending the planner from offline to online planning is possible by applying dynamic updates, e.g., time-variable wind data and remaining battery capacity. Mission planning should generally occur during the time interval of interaction between the UAV and the sensor node to avoid additional strain on onboard computing resources during real-time path and trajectory planning. The interaction time depends on the time required to charge the energy container, such as a battery or supercapacitor. The IPT transmitter described by [15], [16] is capable of powering loads of up to 150 W. Therefore, a small supercapacitor (e.g. 3 Farads) can be rapidly charged within seconds, while charging a Li-ion battery can take tens of minutes.

Our key contributions can be summarized as follows:

- We study autonomous UAV-assisted energy supply management of a WRSN in dynamic environments. We provide a Mixed-Integer Non-Linear Programming formulation for the CSP. To tackle the CSP, we develop an offline and online mission planning scheme and a first-principle power consumption model of different UAV regimes in a windy environment: takeoff, landing, cruise and charging.
- We present ROMP, a novel approach that combines GLS with a Context-aware BHA to generate an initial charging route for the UAV operating in dynamic environments. To our knowledge, this is the first paper to extend classic BHA in addressing the CSP. Our results demonstrate that ROMP can effectively leverage wind conditions to enhance GLS solution performance even in small-scale instances where GLS is highly likely to yield an optimal route. Our results (see §6.5) demonstrated further improvements in ROMP.
- We demonstrate the feasibility of deploying ROMP online, however, the computational time often remains unsatisfactory in practice. We therefore developed and present FastROMP, a novel computationally efficient mission (re-)planning approach. FastROMP incorporates dynamic drop and add operators with CBHA to readjust the previous plan based on the latest parameter updates, omitting the need for re-initialization. Our results (see §6.6) indicate that FastROMP can maximize energy budget utilization and charge more energy with significantly reduced computational time compared to ROMP.
- We present a detailed performance analysis of ROMP and FastROMP under various scenarios. We also examine the parallel search efficiency of CBHA, which provides a general guide for practitioners to apply this planner in ad hoc scenarios.
- Software implementation and experimental results of ROMP are made openly available to the community.1

2 RELATED WORK

Recent advancements in Wireless Power Transfer technology have led to an increased interest in WRSNs. The CSP has emerged as a significant area of research within the field of mission planning for WRSNs. Various studies have employed different modelling strategies and optimization techniques to address the CSP. A review of the relevant literature is presented below.

2.1 Different Modelling Strategies to Address the CSP

Lin et al. modelled a three-dimensional CSP as a TSP with optimization objectives of determining the best charging locations and charging duration [2]. While the TSP model simplifies system complexity and generally converges on relatively low-cost solutions, its uniform charging priority could lead to sensor data loss in practice, particularly in long-term monitoring. Pandiyan et al. addressed a two-stage wireless recharging problem with a weighted clustering GA and the TSP model during the solution search phase [17]. Qian et al. extended this approach by assigning tasks to multiple UAVs, using a K-Means Clustering initialization. This allows a flexible number of vehicle-node visitations in each route [18]. Simulated results showed that their proposed BHA delivered a comparable performance with faster execution than a state-of-the-art GA. However, these clustering strategies did not account for the charging capability of mobile agents, potentially resulting in infeasible missions being assigned to energy-constrained vehicles. Lin et al. proposed an on-demand charging architecture to involve non-deterministic factors in the CSP [19]. Their node deletion and insertion algorithm allowed for flexible adjustment of the current charging queue by avoiding low-efficient nodes and saving dying nodes - similar to the advantages of the OP model.

1. [Online] 1 https://github.com/sysal-bruce-publication/ROMP.git
2.2 Practical Difficulties in UAV Mission Planning

Generating an efficient and safe schedule for a UAV-assisted WRSN is a challenging task. Li et al. addressed this issue by incorporating the UAV’s energy model into their full and partial data collection maximization problem. However, they assumed a fixed hovering and travelling energy consumption rate in the energy model [20]. Thibbotuwawa et al. incorporated weather forecast data into the delivery planning of a UAV fleet, which assumed globally constant wind within a time window [21]. Though the solution in [21] could update online wind speed periodically, it is unreliable because the UAV could not react to a stochastic event such as unforeseen energy deficiency. Dorling et al. employed SA with a simplified energy consumption model for the multirotors to address a dynamic drone delivery problem [22]. Their experimental results highlighted the significance of an appropriate energy model in UAV mission planning. Shi et al. proposed a two-stage approach that combined a UAV Motion Control (UMC) algorithm with a Dynamic Genetic Clustering (DGC) algorithm to search for target sensors and recharge them in unknown environments [23].

The UMC and DGC algorithms were designed to maximize the number of sensors discovered by the UAV without prior location information and to optimize the recharged energy for the WRSN. However, unforeseen adversarial factors during the ‘Power Transfer’ phase were not explicitly considered, potentially resulting in the charging plan generated by the DGC algorithm being infeasible and unnecessary energy waste during the ‘Target Search’ phase. At the time of writing, we are unaware of any other study focused on solving our specific CSP for a single UAV that incorporates dynamic environmental effects into solution performance and mission safety [24], [25], [26]. Some previous research applies approximation algorithms to solve their particular problems. These algorithms can provide a theoretical guarantee by ensuring the solution quality is better than a lower bound. However, the time complexity of approximation algorithms for tackling the dynamic CSP is generally too high for our purposes [2], [27], [28]. We aim to efficiently compute an initial route and optimize its quality for nonlinear system models with online updating capability. Therefore, we apply meta-heuristic algorithms to yield feasible solutions with reasonable latency.

3 System Power Consumption Model

The battery capacity of a UAV restricts its ability to recharge. An appropriate model for UAV energy consumption is essential to evaluate the quantity and distribution of target sensor nodes that need to be covered. In this section, we derive equations for the power consumed by a quadrotor drone during the mission.

The UAV energy consumption can be significantly affected by wind. Integrating wind and UAV aerodynamic models can be computationally expensive, and a high-precision estimate of discharged energy is unnecessary for our planner to obtain a near-optimal solution. Therefore, we employed a model of reduced order complexity. To balance the accuracy and computational complexity of our proposed model, assumptions are made following [29].

1) We discretize the wind field into a time-variant 3D lattice, where each vertex $p_i$ has a wind velocity vector $V_{\text{wind}}(t, p_i)$. Eight adjacent vertices form a wind cube region $\Omega$, whose edge length determines the resolution of the wind field (see Fig. 2). Let $x(t)$ be the coordinate of the UAV at time $t$. We assume the UAV is affected by a constant wind $V_{\text{wind}}(x(t)) = \sum_{i=1}^{8} V_{\text{wind}}(t, p_i)/8$, iff $x(t)$ is in the same region $\Omega$. Therefore, the UAV flight regimes can be approximated to steady-state motion, which mainly considers thrust force, gravity force, and frictional drag force of surrounding airflow [12].

2) The computation will be performed offline in advance, and thus the wind field forecast is reasonably accurate. Because the mission duration is relatively short (typically less than 30 minutes [29], [30]), the magnitude error of actual energy consumption caused by offline and up-to-date wind fields is trivial, and it can be neglected. As the mission progresses, this assumption remains valid on the accumulated error of sensor nodes’ rechargeable energy.

3) We categorize UAV actions during the mission into four distinct regimes: takeoff, landing, cruise, and charging, where takeoff and landing are about vertical movement; cruise is about horizontal movement; The UAV remains static in charging. The power consumption is mainly constituted by induced power $P_i$, profile power $P_{pr}$, parasite power $P_{par}$ and ancillary power $P_a$ for flight regimes [31]. To reduce the non-dominant computation in a dynamic environment (i.e., wind field), we perform a first-principles analysis to estimate $P_i$ and $P_{par}$ only. To estimate the time required for the UAV to reach a target location, we pre-specify a constant UAV ground speed $V_{\text{ground}}$, takeoff speed and landing speed. The UAV traverses from the current node to the next node, following takeoff $\rightarrow$ cruise $\rightarrow$ landing.

4) For the majority of the mission, the UAV is in a steady state, whereby force equilibrium is maintained. Therefore, the vertical thrust is equal to the sum of gravity and shear drag force $F_{T,vert} = F_G + F_{D,vert}$; the horizontal thrust is equal to horizontal drag force $F_{T,h}$. 

---

Fig. 2: An example 2D quiver plot of wind speed over a lake, where the UAV traverses in a wind cube region $\Omega$, formed by eight vertices $A, B, C, D, E, F, G$ and $H$. The wind velocity vector of $\Omega$ is averaged over eight wind velocity vectors at these vertices.
The reference area on the UAV body frame, \( \rho \) is the air density, \( v_a \) is the airflow velocity relative to the propeller, \( \rho_a \) is the gravity force experienced by the PDV, \( g \) is the gravity acceleration, \( A \) is the traverse distance from node \( i \) to \( j \), \( \Omega \) is a set of variables covered by a route, \( \eta_{IPT} \) is the IPT link efficiency for each sensor node may vary depending on factors such as the coil diameter, the distance between coils, and disturbances in the air. Although fluctuations in these parameters can influence overall power consumption and are relevant to our work, they are beyond the scope of this paper. Thus, we set a uniform transfer efficiency to simplify computation. \( E_{IPT} \) can be derived from the energy requirement of a target sensor node:

\[
E_{IPT}(v_i) = \frac{C(V_{max}(v_i) - V_{now}(v_i))^2}{2\eta_{IPT}}
\]

where \( C \) denotes the capacitance, \( V_{max} \) and \( V_{now} \) are the target sensor’s maximum and the present voltage levels.

### 4 Problem Formulation

This paper addresses a dynamic CSP whereby a single UAV is assigned to maintain the power supply of a WRSN in a dynamic environment. The priority for the UAV to recharge a sensor node depends on the sensor node’s remaining energy, whereas a sensor node with lower remaining energy has a higher priority. Because we assume a sensor node’s rechargeable energy is fixed, we use an integer ‘prize’ to normalize all magnitudes. The problem can be defined on a directed complete graph \( G = (V, D, P) \), where \( V = \{v_1, ..., v_N\} \) denotes the set of mission start node \( v_1 \), mission end node \( v_N \) and sensor nodes \( v_j \); \( D = \{d(v_i, v_j), \ldots\} \) refers to the distance set, where \( d(v_i, v_j) \) is the traverse distance from node \( v_i \) to node \( v_j \); \( P = \{p(v_i), ..., p(v_i), ..., p(v_N)\} \) is the set of sensor nodes’ prizes can be collected, where typically \( p(v_1) = p(v_N) = 0 \). Note that we assume each sensor node with a fixed prize during a single mission because the actual magnitude of energy drop is relatively small. For example, the temperature sensor LMT84 has a typical 3.3 \( V \) supply with 5.4 \( \mu A \) power consumption and power-on time of 0.7 \( ms \), which means the energy dissipation will be less than 1 \( \mu J \) for a 30-minute mission if we set the sampling rate a once per minute. Let \( \vec{v}_{ij} \) be the path that the UAV traverses from \( v_i \) to \( v_j \); we then denote the set of wind cube regions covered by a route \( \vec{v}_{ij} \) as \( O(t, \vec{v}_{ij}) = \{\Omega_1(t), ..., \Omega_{ij}(t)\} \), where \( |O(t, \vec{v}_{ij})| \geq 1 \). Because the CSP has decision variable \( X_{ij} \in \{0,1\} \), continuous variable \( t \), and nonlinear constraints, we formulate the CSP to a Mixed-Integer Non-Linear Programming (MINLP) problem as below:
\[
\max \sum_{i=1}^{N-1} \sum_{j=2}^{N-1} p(v_{ij}) \cdot X_{ij}
\]
\[
\text{s.t. } \sum_{j=2}^{N-1} X_{i1} = \sum_{i=1}^{N-1} X_{iN} = 1
\]
\[
\sum_{i=1}^{N-1} \sum_{j=2}^{N-1} \left( \int_{t_{v_i}}^{t_{v_j}} P_f(V_{air}(x(t))) \right) dt + E_{1PT}(v_i) \cdot X_{ij} \leq B, \ x(t) \in O(t, v_1, v_N)
\]
\[
\nu \cdot |t_{v_j} - t_{v_i}| \geq d(v_i, v_j) \cdot V_{ground}^{-1}
\]
\[
lb \leq p(v_i) \leq ub, \ lb, ub \in \mathbb{Z}^+
\]
\[
2 \leq u_i \leq |V|, \ i = 2, ..., N
\]
\[
0 < B \leq \mu \cdot E_{bat}
\]

where the objective function \(6\) maximizes the total collected prize, where the prize has an integer lower bound \(lb\) and upper bound \(ub\) to scale the magnitude difference (see constraint \(13\)). Constraint \(7\) ensures the solution route starts at node \(v_1\) and ends at node \(v_N\). Constraint \(8\) ensures the connectivity of the solution route, and one node is visited at most once. Constraint \(9\) and \(14\) prevent subtours, according to the Miller-Tucker-Zemlin formulation of the TSP \(33\). Constraint \(10\) limits the total energy consumption is less than the energy budget \(B\). Note that \(E_{1PT}(v_i)\) can be calculated from Eq. \(5\), and \(P_f\) can be obtained from Eq. \(4\) depending on different flight regimes. Constraint \(11\) enforces the UAV air velocity to be static (i.e., steady state) iff the UAV traverses in the same wind cube region \(\Omega\). Constraint \(12\) keeps track of the elapsed time from the UAV leaves node \(v_i\) to the UAV leaves node \(v_j\). Constraint \(15\) is set for safety reasons (the factor \(\mu\) is typically set to 80%), which ensures the UAV has enough energy to execute the Return-To-Home process.

The above MINLP formulation can be simplified as below:

\[
\min \sum_{i=1}^{N-1} \sum_{j=2}^{N-1} \left( \int_{t_{v_i}}^{t_{v_j}} P_f(V_{air}(x(t))) \right) dt \cdot X_{ij}
\]
\[
\text{s.t. } \sum_{j=2}^{N-1} X_{i1} = \sum_{i=1}^{N-1} X_{iN} = 1
\]
\[
\sum_{i=1}^{N-1} \sum_{j=2}^{N-1} X_{i1} = \sum_{i=1}^{N-1} X_{iN} = 1
\]
\[
u_i - u_j + 1 \leq (N - 1)(1 - X_{ij}), \ i, j = 2, ..., N - 1
\]
\[
2 \leq u_i \leq |V|, \ i = 2, ..., N - 1
\]

Constraints \(11, 12\)

where the objective function \(16\) minimizes the energy consumption caused by flight regimes only. Constraint \(17\) represents that the route starts and ends at the same node. Constraint \(18\) enforces all sensor nodes are visited exactly once. Constraint \(19\) and \(20\) prevent subtours.

### 5 System Design

To simulate a general mission planning scenario of UAV-assisted WRSN, we select temperature sensor LMT84 \(32\) with capacitor HB1030-2R510 \(34\) and pressure sensor NPA300 \(35\) with capacitor PHB-5R0 \(36\) as the sensor prototypes, following \(17, 18\). Specifications of sensor nodes and the UAV are listed in Table. \(2\). Values of some parameters in Eq. \(1, 5\) that experiments can identify are summarized as below: air density \(\rho = 1.25 kg/m^3\), drag coefficient \(C_D = 0.04\) \(31\), and IPT link efficiency \(\eta_{1PT} = 50\%\) \(17\). Although fluctuations in these parameters can influence overall power consumption and are thus relevant to our work, they are beyond the scope of this paper. Therefore, we set constant values to these parameters.

#### 5.1 System Architecture

We introduce a two-stage, offline and online, mission planning scheme to tackle the CSP. The workflow is visualized in Fig. \(4\). For a specific number of charging requests, the base station first conducts an offline mission planning phase, allocating a single UAV to complete the mission autonomously. There are various methods to make the system continuously autonomous. For example, a UAV can be deployed to collect the initial energy distribution of the WRSN and the energy decay rate of each sensor node.
The UAV may conduct a routine online mission planning, utilizing the time taken for charging a sensor node to verify that it can complete its mission. Alternatively, if the UAV’s remaining energy level is approaching a safety threshold value, it may also trigger the online scheme.

5.2 Rapid Online Metaheuristic-based Planner (ROMP)

5.2.1 ROMP for Offline Mission Planning Scheme

The base station, which is assumed to have adequate computing resources, conducts the offline scheme before the mission starts to assign a suitable charging plan for the UAV. The offline ROMP consists of an ‘Initial Solver’ and an ‘Optimization Solver’, whose architecture is visualized in Fig. 5a. The planner inputs are Sensor Node Data (SN-DATA) which includes sensor node coordinates, sensor type, capacitor voltage level and associated prize, and UAV Data (UAV-DATA) which includes UAV’s present coordinate, mission end coordinate and UAV remaining battery.

The ‘Optimization Solver’ further optimizes the initial solution in the given dynamic simulation environment. After $N_{gen}$ generations of evolution, where $N_{gen}$ is user-defined, the ‘Optimization Solver’ will return the best-known solution to the UAV. We present the main algorithm, Context-aware Black Hole Algorithm (CBHA), in the ‘Optimization Solver’ as follows. The BHA, first presented in [38], is a metaheuristic optimization algorithm inspired by the black hole phenomenon. In BHA, the solutions to the optimization problem are represented as particles in a search space, and the best available solution is characterized as the black hole. The black hole has a probability $P_a$ to attract other solutions towards itself, causing stochastic displacements in the search space. CBHA mainly comprises three phases: initialization, attraction, and absorption.

The initialization phase generates a set of populations from an initial route, utilizing a probabilistic candidate search operator (see Alg. [I]). This operator is typically more efficient than the original version because it considers the influence of index permutation on the fitness metric. It is applied to each element in the initial route. Moreover, to filter out some visit that has a low prize or that are too far from the black hole route, we employ a simplified fitness metric $M$ that is based on the fixed distance between sensor nodes and their corresponding prizes:

$$M(v_j) = W_{re} \cdot \left( w(v_j) - l_b \right) - W_{de} \cdot l_g \left( d(v_{i-1}, v_j) + d(v_j, v_{i+1}) \right)$$  \hspace{1cm} (21)

where $W_{re}$ and $W_{de}$ are two weight coefficients for recharged and discharged energy separately; $d(v_{i-1}, v_j)$ (or

Algorithm 1: Candidate search operator

| Input | Sensor node set $V$, route $\{v_1, ..., v_N\}$, target node $v_i$ in route, search number $N_{srch}$ |
|-------|---------------------------------------------------------|
| Output| A candidate list of length $N_{srch}$                   |

1 Record previous node $v_{i-1}$ and next node $v_{i+1}$ of target node $v_i$ in route;
2 for node $v_k \in V \setminus \{v_1, v_N, v_{i-1}, v_{i+1}\}$ do
3 Compute fitness metric $M(v_k)$ (Eq. (21));
4 Append the pair $(M(v_k), v_k)$ to a list $L$;
5 Order $L$ with descending order;
6 ∀ pair $(M(v_k), v_k) \in L$, sequentially add $v_k$ into candidate list until its length equals to $N_{srch}$;
7 return Candidate list.
Algorithm 2: Attraction phase

Input: \( Route \{ v_1, ..., v_i, ..., v_N \} \),
black hole route \( v_i^{BH} \{ v_1, ..., v_i^{BH}, ..., v_N \} \)

Output: A modified route after attraction phase

1. for \( i = 2 \) to \( N_{route} - 1 \) do
2.     if \( random[0,1] \leq P_a \) and \( v_i \neq v_i^{BH} \) then
3.         Compute the new coordinate \( x'_i \), using \( v_i \) and \( v_i^{BH} \) (Eq. (22));
4.         For \( x'_i \) find a closest node \( v'_i \) \( \in \mathcal{V} \) \( \setminus \{ v_1, v_N \} \);
5.         If \( v'_i \in route \), swap \( v_i \) and \( v'_i \);
6.         Otherwise, \( v_i \leftarrow v'_i \);
7.     return New route.

Algorithm 3: ROMP for online mission planning

Input: SN-DATA, UAV-DATA, wind field
Output: Black hole route \( v_i^{BH} \{ v_1, ..., v_i^{BH}, ..., v_N \} \)

1. GLS solves with full prize budget \( B = p_{sum} \);
2. Obtain the initial route and evaluate its energy cost;
3. Execute the step reduction loop to obtain the first feasible route \( \{ v_1, ..., v_N \} \);
4. Initialize \( N_{pop} \) populations from route and update their fitness metric values (Eq. (23));
5. while not reach stopping criteria do
6.     for \( route \in populations \) do
7.         Perform attraction phase (Alg. 2) to route;
8.         Compute the fitness metric of route;
9.     if new fitness metric is better then
10.        Update new route and its fitness metric;
11.    Update black hole route \( v_i^{BH} \) and its fitness metric;
12.    Compute event horizon radius \( r \);
13.    for any population fitness metric < \( r \) do
14.        Perform absorption phase to population route;
15.        Update new route and its fitness metric;
16.    Update black hole route \( v_i^{BH} \) and its fitness metric;
17. return Black hole route \( v_i^{BH} \).

\( d(v_j, v_{j+1}) \) is the distance between the sensor node \( v_j \) and the previous (or next) sensor node of \( v_i \) in the target solution \( S_t \). We subtracted \( w(v_j) \) by \( lb \) and applied a common logarithm to restrict these two factors within a similar magnitude scale because the prize ranges from \( lb \) to \( ub \), and distance ranges from meters to thousands of meters. CBHA is designed to prioritize the search for potential sensor nodes with higher prizes, resulting in lower priority assigned to sensor nodes around \( lb \).

The attraction phase (Alg. 2) facilitates search space exploration by attracting candidate routes towards promising regions. The attraction of a node at index \( i \) of a candidate route is formulated as:

\[
x_i' = x_i + \alpha \cdot (x_i^{BH} - x_i), \quad \forall i \in 2, ..., N_{route} - 1 \tag{22}
\]

where \( x_i \) and \( x_i^{BH} \) are the coordinates of the node at index \( i \) of a candidate route and a black hole, respectively. We then approximate the new coordinate \( x_i' \) to the closest node in the feasible sensor node set \( \mathcal{V} \) to complete this process.

To maintain the diversity of the population and avoid stagnation at local extremes, the absorption phase re-initializes (same as the initialization phase) the population solution whose fitness metric is within the event horizon radius \( r \). Moreover, to align with the context of CSP, we formulate the fitness metric of the entire route as follows:

\[
M = \text{penalty} + \sum_{i=1}^{N_{route}} E_{re} + \sum_{i=1}^{N} E_{re} + \sum_{i=1}^{N_{route}} E_{ipt} + \int_{t_{v_i}}^{t_{v_{N}}} P_f dt \tag{23}
\]

where the penalty is \(-100\) when the route cannot satisfy Constraint (10), otherwise the penalty is 0; the second term refers to the ratio of UAV recharged energy to all rechargeable energy of the WSN; the third term denotes the ratio of UAV discharged energy to UAV initial energy.

Parallelization of serial CBHA can be achieved with Message Passing Interface (MPI) because population solutions are generated from a common source, enabling uniform initialization and evolution process distribution. Each process’s black hole route is collected periodically (determined by aggregate frequency) to update a global black hole route.

5.2.2 ROMP for Online Mission Planning Scheme

With minor modifications, ROMP can be employed for online mission planning. We present the pseudo-code of online ROMP as Alg. 3. To assess the performance of ROMP, a comprehensive analysis is conducted in § 6.6. Our results indicate that, in specific large-scale scenarios, the step reduction loop in the ‘Initial Solver’ can be highly time-consuming. This means ROMP may exhibit limited performance when applied to online (re-)planning scenarios.

5.3 FastROMP

To further decrease the computational complexity of ROMP, we have developed FastROMP. This approach incorporates online updates such as UAV flight time, remaining battery level and mission progress into the planning process. This allows for dynamic adjustments to be made to the previously established route. Drawing inspiration from [39], we have designed an evaluation metric for dropping or adding a sensor node within the context of the CSP. The architecture of FastROMP is shown in Fig. 5b. FastROMP begins by calibrating the route yielded from the previous computation to ensure its accuracy. If the solution cannot comply with the energy budget constraint, ROMP invokes the drop operator iteratively until a feasible solution is reached. We define a metric, drop value, to evaluate the behaviour of dropping one sensor node \( v_i \) (if removed at route index \( i \)):

\[
drop(v_i) = \frac{\int_{t_{v_i}}^{t_{v_{i+1}}} P_f dt + \int_{t_{v_i}}^{t_{v_{i+1}}} P_f dt - \int_{t_{v_{i-1}}}^{t_{v_i}} P_f dt}{\int_{t_{v_{i-1}}}^{t_{v_i}} P_f dt} \tag{24}
\]

where the denominator term refers to the discharged energy difference if removed node \( v_i \) from the route. \( v_{i-1} \) and \( v_{i+1} \) denote the previous node and next node in the route, respectively. Alg. 4 demonstrates the drop operator algorithm. Subsequently, the add operator is repeatedly applied until the solution attains maximal utilization of the allocated energy budget. Specifically, the add operator (Alg. 5) finds
the best position for inserting the unvisited node. We define
the add value of a sensor node $v_i$ (if inserted at route index $i$) as:
\[
\text{add}(v_i) = \begin{cases} 
\text{Eq. (24)} & \text{if added satisfy Constraint (10)} \\
0 & \text{otherwise}
\end{cases}
\]

We used the same metric formulated for the drop value when
the route is feasible after insertion. However, to guarantee
the feasibility of the solution for further optimization, it is necessary to re-simulate the visit of any
newly added/dropped sensor node at route index $i$ and all
subsequent visits. To better align with the context of
the dynamic CSP, we assign two lists to keep track of the energy
consumption and the departure time for each visit.

Alg. 6 shows the pseudo-code of FastROMP. Since the
greedy drop and add operators can drive the previous route
towards local optimum, we use CBHA to explore the potential
for finding a global optimal route. CBHA’s stopping
criteria can be determined by one or more factors, such as the

6 EXPERIMENTS, RESULTS AND DISCUSSION
This section presents numerical results to evaluate offline
and online mission planning schemes. The planner was
coded in C++20 and executed on a 64-bit Intel Core i7-8550U
CPU running at 1.80 GHz with 8 GB RAM (i.e., DJI Manifold
2-C [40], a typical onboard flight computer). The following
attributes (Attrs) were considered when evaluating the sol-
solution performance:

\[ t \text{ Algorithm execution time [unit: s]} \]
\[ E^\star_v \text{ Recharged energy of visited sensor nodes in a single mission [unit: J]} \]
\[ E^\star_u \text{ UAV discharged energy in a single mission [unit: Wh]} \]
\[ R_{te} \text{ The ratio of total recharged energy to all rechargeable energy of requested sensor nodes [unit: %]} \]
\[ R_{de} \text{ The ratio of total UAV discharged energy to UAV initial energy [unit: %]} \]
\[ R_{rd} \text{ The conversion efficiency from the UAV’s discharged energy to sensor node recharged energy [unit: %]} \]
Parallelization performance summary. The probability is the percentage that CBHA obtains a solution better than GLS over all executions. The 4-process execution is capable of achieving 100% convergence at 5000 populations, while the serial execution fails to converge even with a population size of 10000.

Empirical cumulative distribution function for finding better solutions against time using ROMP. The results span 500 executions using $N_{\text{pop}} = 5000$ and $N_{\text{gen}} = 100$. The 4-process execution can save $\sim 30\%$ execution time over serial while achieving better solution quality.

Fig. 6: Parallel search performance analysis

where:

\[
R_{re} = \left( \frac{E^*_{re}}{\sum_{i=2}^{N-1} E_{re}(v_i)} \right) \cdot 100\% \tag{26}
\]

\[
R_{de} = \left( \frac{E^*_{de}}{B} \right) \cdot 100\% \tag{27}
\]

\[
R_{rd} = \left( \frac{E^*_{re} - (3600 \cdot E^*_{de})}{3600 \cdot E^*_{de}} \right) \cdot 1000\% \tag{28}
\]

We set the lower prize bound $lb$ to 6 and upper bound $ub$ to 10 to distinguish between relatively discharged (6-10) and charged (1-5) sensor nodes and to reduce the number of sensor nodes involved in the computation. Arguments in the algorithm can be tuned according to different requirements. Setting a large population and generation number can, for example, lead to deep searches and long computation time. In CBHA, we set the population number $N_{\text{pop}}$ to 80, generation number $N_{\text{gen}}$ to 80, ‘attraction’ probability $P_a$ to 75\%, search number $N_{\text{srch}}$ to 10, event horizon radius $r$ at 25\%, and aggregate frequency as 10 as a common setting. The parameter settings for $P_a$, $N_{\text{srch}}$ and $r$ are selected to allow the algorithm to evolve more actively. It is important to note that updating the global solution should not frequently occur since this process involves collecting routes from other processes, sorting fitness metrics, broadcasting a global route and updating the black hole route, which can have a non-trivial impact on the execution time. The update frequency represents a trade-off between computing time and search efficiency. These chosen values were based on a scenario where a UAV requires tens of seconds to charge a supercapacitor. Thus, ROMP will produce a safe route, but it may not be more efficient than GLS. A detailed analysis of $N_{\text{pop}}$ and $N_{\text{gen}}$ setting can be seen in § 6.2, which shows that setting the population to 5000 allows our algorithm to produce a better solution than GLS in 100\% of cases tested.

6.1 Meta-heuristic Algorithm Options in OR-Tools

OR-Tools supports different local search options: AUTO, GD, GLS, SA, and TS [37]. These options were tested and averaged over 20 executions (see Fig. 7). AUTO and GD completed the calculation quickly but yielded solutions with the lowest fitness metric. Note that the solution provided by OR-Tools is a basis of CBHA’s calculation, which means a low-quality initial solution will increase the opportunity for the CBHA to find a better solution, which is not helpful for performance analysis. Therefore, we selected GLS as the benchmark algorithm, as it found the solutions with the highest fitness metric in an acceptable computation time.

6.2 Parallel Search Efficiency of the CBHA

Parallelization performance is analyzed based on the OP scenario, which aims at obtaining the probability that the ‘Optimization Solver’ finds a route with a better fitness value than the one from GLS, with varying numbers of processes over which computation can be distributed. Because CBHA is a population-based heuristic algorithm, the number of generations has little impact on performance.
We therefore fixed the generation number to 100 for these experiments. Fig. 6a demonstrates the parallelization performance with constraints of increasing population numbers over 100 executions. A logistic function curve was fitted to the data points to show the trend with increasing populations. Results indicate that the serial execution’s probability of finding a better solution cannot converge to 100% even using 10000 populations, while the 4-processes execution converges to 100% at 5000 populations. For a small population number, e.g., $N_{\text{pop}} \leq 500$, parallel execution achieves better solutions with approximately linear probability improvement. The execution time for the ‘optimization solver’ to achieve a trusted solution can therefore be reduced linearly with increasing allocated processes. To find the minimum execution time needed to gain a better solution than GLS, another experiment was conducted using 5000 populations, and 100 generations, in which the 4-process execution can converge to 100%. For a small population number, e.g., $N_{\text{pop}} \leq 500$, parallel execution achieves better solutions with approximately linear probability improvement. The execution time for the ‘optimization solver’ to achieve a trusted solution can therefore be reduced linearly with increasing allocated processes. To find the minimum execution time needed to gain a better solution than GLS, another experiment was conducted using 5000 populations, and 100 generations, in which the 4-process execution can converge to 100%. Fig. 6a demonstrates the empirical cumulative distribution (ECDF) of the execution time to achieve a better solution. With 4 allocated processes, the CBHA needs at least 5 minutes to guarantee a solution of better performance than OR-Tools. But compared to serial execution, the 4-process execution can save computation time by ~30% of the total execution time. Among all cases where CBHA achieved better solutions, using four processes had a smaller mean (112.04 s) and Standard Deviation (SD, 45.71) of execution time compared to serial execution (mean 149.76 s, SD 81.29). Although a simple strategy was used in this experiment, the results clearly show that parallelization can enhance the solver’s stability and reduce the computational time needed for convergence. The experiment also proves that ROMP can be used for online re-planning on regular flight computers.

### 6.3 Wind Impact on Solution Performance

We selected the TSP scenario to explore the effect of wind on the ‘Optimization Solver’ performance because additional energy consumption caused by wind (instead of other reasons like sequence order change) can be easily observed. For instance, in the last iteration of case 140 in Table 3, ROMP found a solution that saved 0.330 Wh whilst a north-east 3.606 m/s constant wind was present by reversing the visiting sequence of the initial solution. Another interesting comparison, where we tested 100 TSP cases with random sensor node deployments, is shown in Fig. 9, where we observe a 0.112 Wh reduction in discharged energy under a constant westerly 5 m/s wind.

**TABLE 3: System performance in OP-1 and OP-2 scenarios**

| Attr  | Cases | OP-1   | OP-2   |
|-------|-------|--------|--------|
| $E_{\text{de}}$ | 540.71 | 224.77 |
| $R_{\text{de}}$ | 59.382 | 52.721 |
| $E_{\text{de}}$ | 71.879 | 68.594 |
| $R_{\text{de}}$ | 71.879 | 68.663 |
| $R_{\text{rd}}$ | 2.0917 | 0.9102 |
| $W_{\text{re}}$ | 20 | 50 | 80 | 20 | 50 | 80 |
| $E_{\text{re}}$ | 538.70 | 543.39 | 545.94 | 211.63 | 232.60 | 263.00 |
| $R_{\text{re}}$ | 59.163 | 59.679 | 59.953 | 49.653 | 54.377 | 61.679 |
| $E_{\text{re}}$ | 65.745 | 71.279 | 71.721 | 59.077 | 64.123 | 75.592 |
| $R_{\text{re}}$ | 65.810 | 71.350 | 71.793 | 59.137 | 64.187 | 75.668 |
| $R_{\text{rd}}$ | 2.2749 | 2.1174 | 2.1114 | 0.9915 | 1.0068 | 0.9674 |

![Fig. 8: An OP-1 instance of GLS’ and ROMP’s solutions using different search strategies.](image)

![Fig. 9: An instance showing changes in route sequence and discharged energy after applying a constant west wind.](image)
In this section, we focus on OP scenarios, given the acceptable solution quality of GLS for TSP scenarios. Two scenarios were tested in our experiments: OP-1, a scenario with many sensor nodes randomly deployed in a middle-scale area size (40 sensor nodes in a $2500 \times 2500$ m$^2$ grid), and OP-2, a scenario with few nodes randomly deployed in a large-scale area (20 sensor nodes in a $4000 \times 4000$ m$^2$ grid). Note that although the attribute of every sensor node is generated randomly in each case, the coordinates of all sensor nodes remain the same in each OP-1 and OP-2 experiment. This is to test the sensitivity of the planner to prize changes. We consider three search strategies: ‘Charge-More’ ($W_{re} = 80$), ‘Balance’ ($W_{re} = 50$), and ‘Save-Energy’ ($W_{re} = 20$). Fig. 8 shows a visualization of typical results using different search strategies. For example, compared with the GLS’s route, the ‘Charge-More’ strategy replaces a temperature sensor node with prize 9 in the middle with the GLS’s route, which leads to an extra charging of 18.19 J energy.

To assess ROMP’s performance in general scenarios, we demonstrate its averaged execution results in Table 4, among 300 randomized cases, where highlights in bold indicate distinctly improved solutions, relative to an initial solution and highlights in italics and bold indicate the best solution. The search strategy’s impact on results varies depending on the grid scale and sensor node density. For example, in OP-2, the ‘Charge-More’ strategy resulted in ~10% more network energy charged than the ‘Save-Energy’ strategy, while in OP-1, the difference is less than 1%. An increasing trend of recharged energy ratio $R_{re}$ can be observed with increases of $W_{re}$ and $R_{de}$. Because of the biased penalty design, GLS tends to first add more sensor nodes with higher prize weight into the solution list. The solution length, discharged energy, and recharged energy may be limited because the collected prize reaches the threshold value of ‘capacity’ quickly during searching. This can explain why UAV discharged energy is ~70% in most OP cases.

### 6.4 ROMP Performance in OP Scenarios

In this section, we focus on OP scenarios, given the acceptable solution quality of GLS for TSP scenarios. Two scenarios were tested in our experiments: OP-1, a scenario with many sensor nodes randomly deployed in a middle-scale area size (40 sensor nodes in a $2500 \times 2500$ m$^2$ grid), and OP-2, a scenario with few nodes randomly deployed in a large-scale area (20 sensor nodes in a $4000 \times 4000$ m$^2$ grid). Note that although the attribute of every sensor node is generated randomly in each case, the coordinates of all sensor nodes remain the same in each OP-1 and OP-2 experiment. This is to test the sensitivity of the planner to prize changes. We consider three search strategies: ‘Charge-More’ ($W_{re} = 80$), ‘Balance’ ($W_{re} = 50$), and ‘Save-Energy’ ($W_{re} = 20$). Fig. 8 shows a visualization of typical results using different search strategies. For example, compared with the GLS’s route, the ‘Charge-More’ strategy replaces a temperature sensor node with prize 9 in the middle with another pressure sensor node with prize 9 in the middle right, which leads to an extra charging of 18.19 J energy.

To assess ROMP’s performance in general scenarios, we demonstrate its averaged execution results in Table 4, among 300 randomized cases, where highlights in bold indicate distinctly improved solutions, relative to an initial solution and highlights in italics and bold indicate the best solution. The search strategy’s impact on results varies depending on the grid scale and sensor node density. For example, in OP-2, the ‘Charge-More’ strategy resulted in ~10% more network energy charged than the ‘Save-Energy’ strategy, while in OP-1, the difference is less than 1%. An increasing trend of recharged energy ratio $R_{re}$ can be observed with increases of $W_{re}$ and $R_{de}$. Because of the biased penalty design, GLS tends to first add more sensor nodes with higher prize weight into the solution list. The solution length, discharged energy, and recharged energy may be limited because the collected prize reaches the threshold value of ‘capacity’ quickly during searching. This can explain why UAV discharged energy is ~70% in most OP cases.

### 6.5 ROMP Performance with Increasing Nodes

A functional requirement of our mission planner is to charge an entire network using a single UAV. A network consisting of 10 to 150 sensor nodes in a $2000 \times 2000$ m$^2$ grid is used to evaluate our system’s performance, where the attribute of each sensor node is randomized. System performance is indicated by the number of UAV mission iterations $N_{iter}$ required to recharge the entire network and the attributes listed in Table 4. Note that in this experiment, we did not use metrics of $R_{re}$ and $R_{de}$ because the UAV may not have enough unvisited sensors in some missions, which leads to an unfair comparison of $R_{re}$ and $R_{de}$. All execution results are averaged over 20 executions. GLS performance degrades as network scale increases, which leaves more space for ROMP to improve. An example can be found in case 150, where the results show that ROMP successfully improves
the solutions provided by GLS with an average of 38.01 J more recharged energy and 20.079 Wh less discharged energy in total. Fig. 10 shows the ROMP’s routes to charge 100 sensor nodes, requiring four missions (Fig. 10a to Fig. 10d).

6.6 System Performance with Dynamic Updates of the UAV Remaining Energy

To simulate scenarios where a UAV encounters an unpredictable force, such as an unexpected headwind that causes its remaining energy to be depleted at a faster rate, we set the UAV’s remaining energy to be arbitrarily reduced to a sufficiently low level whilst it is executing its route. Fig. 11 shows how ROMP adjusts the UAV’s route. The panels from left to right illustrate the original route, the first updated route after the UAV’s remaining energy is reduced, the second updated route after a further reduction in remaining energy, and the final route. The first change occurs once the UAV has charged five sensor nodes, after which its remaining energy is reduced to 50 Wh. The second change occurs once the UAV has recharged two additional sensor nodes, after which its remaining energy is further reduced to 22 Wh. In this case, the UAV may not have enough energy to execute its original route. The planner, therefore, adjusts the estimated mission execution capability of the UAV, which yields a route of 16.18 Wh discharged energy.

Accordingly, we propose FastROMP that eliminates the need to re-execute the ‘Initial Solver’ by directly adjusting the previous route. To evaluate the proposed scheme, three distinct cases (case 50, 100, and 150 from § 6.5) were selected to represent the sparsely, moderately and densely distributed deployment of a WRSN, respectively. All experimental results were averaged over 20 executions. Fig. 13 illustrates the variations in the UAV’s discharged and charged energy in response to two scenarios wherein the UAV charged ten sensor nodes, and its actual battery was 10% lower (and higher) than the expected level. The expected battery levels of the three cases considered here were 76.4435 Wh, 82.7459 Wh, and 82.3198 Wh, respectively. Note that though case 100 and case 150 have different sensor densities, their expected battery levels are similar. This is because both cases are close to the upper bound of UAV discharged energy (for charging ten nodes in this grid size). Results
indicate that ROMP can offer better solution quality than FastROMP when the UAV can charge all remaining sensor nodes. For instance, in case 50 of Fig. 13b, FastROMP’s route has similar recharged energy as the ROMP’s but more discharged energy (1.59%) than ROMP’s. While in other cases, FastROMP effectively utilized the energy budget to improve the solution quality. Notably, in case 100 of Fig. 13a, FastROMP’s route can obtain a gain of recharged energy (21.47%) with a minor cost of UAV discharged energy (1.89%), compared to ROMP’s route. Moreover, as presented in Fig. 13 FastROMP significantly reduces computation time by employing an online adjustment on the previous route. Depending on scenarios of whether the energy budget decreased (Fig. 13a) or increased (Fig. 13b), FastROMP demonstrated an increasing trend in saved execution time, ranging from 39.57% to 93.3%. This corresponds to a sensor node deployment from sparse to dense distribution.

### 6.7 Complexity Analysis

In the described tests, a two-dimensional vector with memory size \((N_{pop} \cdot N_{route})\) is used for recording population solutions. As for the time complexity of CBHA, it takes \((N_{route} \cdot N_{sn})\) iterations to complete the ‘attraction’ phase, which is the most computationally demanding function in CBHA’s evolution loop. The worst-case iterations taken for the main loop, therefore, should be \((N_{sol} \cdot N_{sn} \cdot N_{gen} \cdot N_{pop})\). Note that \(N_{sol} = N_{sn}\) in TSP scenarios, which takes \((N_{sn}^2 \cdot N_{gen} \cdot N_{pop})\) iterations to compute. Therefore, CBHA has a time complexity of \(O(n) \sim O(n^2)\) when \(N_{sn}\) is considered the main input. For FastROMP, both operators require three vectors, each with a memory size \(N_{route}\), to keep track of the UAV’s route, energy, and flight time. During the computation of the add value list, the add operator requires a maximum of \( ((N - N_{route})^2 \cdot N_{route})\) iterations, which is the most computationally expensive part of the algorithm. The drop operator needs at most \(N_{route}^2\) iterations to calculate the drop value list. As a result, the sequential execution of both operators will lead the FastROMP’s worst-case time complexity to be \(O(n^2)\).

### 7 Conclusion and Future Work

This paper presents a lightweight and reliable planner, ROMP, to solve the charging scheduling problem for a wireless rechargeable sensor network system. Characterized as a hybrid of dynamic TSP and OP scenarios, the optimization objective is to maximize the recharged energy of sensor nodes within the UAV energy constraint. We first develop an energy consumption model for M100 drones that considers flight regimes in the wind and IPT. The offline mission planning scheme involves an ‘Initial Solver’ and an ‘Optimization Solver’, which employs a Context-aware BHA to improve the initial solution from GLS. Implementing a straightforward parallelization strategy exploiting multiple processing cores further improves the performance of the offline scheme. Results show that using four processes, our distributed algorithm can achieve better solutions with a percentage of 99.8% (over 500 executions) than GLS at 5000 populations. More importantly, it demonstrates that the system can converge in sufficient time for deployment in a realistic scenario, as shown on a typical flight computer. Results indicate that even in a TSP scenario, the high-quality solutions provided by GLS can be further improved by ROMP in windy environments. However, the step reduction loop in ‘Initial Solver’ is computationally expensive and may not be applicable in the real-world scenario in time. Therefore, we developed FastROMP, which employs
a new online adjustment operator that takes the latest state information as input. Exploiting the prior route from the last state, FastROMP is shown to deliver updated routes that better utilize the UAV’s energy budget and charge more sensor nodes with a significant computational time reduction, compared to ROMP.

Our future work will focus on improving the reliability and efficiency of ROMP and demonstrating the efficacy of the planner using UAV and sensor hardware at the appropriate scale. Field and wind tunnel experiments will be conducted to achieve robust performance and algorithm parameters. We will expand model parameters to include dynamic estimates of drag coefficient, fluid mass density, tilt angle changes, etc. We will also investigate the use of machine learning techniques, such as Long Short-Term Memory (LSTM), to predict energy consumption. By training a power consumption model with a data-driven LSTM neural network, we may reduce the computational expense for the onboard planner [41]. Moreover, we will explore the simulation of various event-based sensor types in WRSNs, including motion and audio sensors. These sensors typically exhibit a stochastic energy depletion rate, which is contingent upon the distribution of detected events. This characteristic can result in a stochastic ‘prize’ associated with visiting a node. Therefore, the scheduling problem presents a compelling opportunity for robust optimization.

References

[1] Z. Fan, Z. Jie, and Q. Yujie, “A survey on wireless power transfer based charging scheduling schemes in wireless rechargeable sensor networks,” in 2018 IEEE 4th International Conference on Control Science and Systems Engineering (ICCSSE), pp. 194–198, 2018.

[2] C. Lin, W. Yang, H. Dai, T. Li, Y. Wang, L. Wang, G. Wu, and Q. Zhang, “Near optimal charging schedule for 3d wireless rechargeable sensor networks,” IEEE Trans. on Mobile Computing, pp. 1–1, 2021.

[3] P. D. Mitcheson, D. Boyle, G. Keliris, D. Yates, J. A. Saenz, S. Aldhafer, and E. Yeatman, “Energy-autonomous sensing systems using drones,” in 2017 IEEE SENSORS, pp. 1–3, 2017.

[4] D. E. Boyle, M. E. Keliris, P. D. Mitcheson, and E. M. Yeatman, “Energy provision and storage for pervasive computing,” IEEE Pervasive Computing, vol. 15, no. 4, pp. 28–35, 2016.

[5] P. Mitcheson, G. Keliris, S. Aldhafer, J. Arteaga, D. Yates, D. Boyle, and E. Yeatman, “Power electronics for wireless power delivery in synthetic sensor networks,” Journal of Physics: Conference Series, vol. 1052, p. 012145, 07 2018.

[6] M. Fischetti, J. J. S. Gonzalez, and P. Toth, “Solving the orienteering problem through branch-and-cut,” INFORMS Journal on Computing, vol. 10, no. 2, pp. 133–148, 1998.

[7] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman, “Learning combinatorial optimization on graphs: A survey with applications to networking,” IEEE Access, vol. 8, pp. 120388–120416, 2020.

[8] H. Awad and R. Elshaker, “A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants,” Computers & Industrial Engineering, p. 106242, 12 2019.

[9] A. Hatamlou, “Solving travelling salesman problem using black hole algorithm,” Soft Computing, vol. 22, 12 2018.

[10] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, and D. Wu, “Offline and online search: Uav multipath flight planning under dynamic urban environment,” IEEE Internet of Things Journal, vol. 5, no. 2, pp. 546–558, 2018.

[11] Y. Wang, J. O’Keeffe, Q. Qian, and D. Boyle, “Kinojgm: A framework for efficient and accurate quadrotor trajectory generation and tracking in dynamic environments,” 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 310–315, 2017.

[12] Y. Chen, D. Baek, A. Bocca, A. Macii, E. Macii, and M. Poncino, “A case for a battery-aware model of drone energy consumption,” in 2018 IEEE International Telecommunications Energy Conference (INTELEC), pp. 1–8, 2018.

[13] Z. Liu, R. Sengupta, and A. Kurzhanskiy, “A power consumption model for multi-rotor small unmanned aircraft systems,” in 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 310–315, 2017.

[14] T. Rodrigues, J. Patrikar, A. Choudhry, J. Feldgoise, V. Arcot, A. Gahlaut, S. Lau, B. Moon, B. Wagner, H. Matthews, S. Scherer, and C. Samaras, “In-flight optical and energy use data set of a dji matrice 100 quadcopter for small package delivery,” Scientific Data, vol. 8, 06 2021.

[15] J. M. Arteaga, S. Aldhafer, G. Keliris, C. Kwan, D. C. Yates, and P. D. Mitcheson, “Dynamic capabilities of multi-mhz inductive power transfer systems demonstrated with batteryless drones,” IEEE Trans. on Power Electronics, vol. 34, no. 6, pp. 5093–5104, 2019.

[16] S. Aldhafer, P. D. Mitcheson, J. M. Arteaga, G. Keliris, and D. C. Yates, “Light-weight wireless power transfer for mid-air charging of drones,” in 2017 11th European Conference on Antennas and Propagation (EUCAP), pp. 336–340, 2017.

[17] A. Y. S. Pandiyan, D. E. Boyle, M. Kizimiglo, S. Wright, and E. Yeatman, “Optimal dynamic recharge scheduling for two stage wireless power transfer,” IEEE Trans. on Industrial Informatics, pp. 1–1, 2020.

[18] Q. Qian, A. Y. S. Pandiyan, and D. E. Boyle, “Optimal recharge scheduler for drone-to-sensor wireless power transfer,” IEEE Access, vol. 9, pp. 59301–59312, 2021.

[19] C. Lin, J. Zhou, C. Guo, H. Song, G. Wu, and M. S. Obaidat, “Tscr: A temporal-spatial real-time charging scheduling algorithm for on-demand architecture in wireless rechargeable sensor networks,” IEEE Transactions on Mobile Computing, vol. 17, no. 1, pp. 211–224, 2018.

[20] M. F. Sohail, C. Y. Leow, and S. Won, “Energy-efficient non-orthogonal multiple access for uav communication system,” IEEE Trans. on Vehicular Technology, vol. 68, no. 11, pp. 10834–10845, 2019.

[21] A. Thibbotuwawga, G. Bocewicz, P. Nielsen, and B. Zbigniew, “Planning deliveries with uav routing under weather forecast and energy consumption constraints,” IFAC-PapersOnLine, vol. 52, no. 13, pp. 820–825, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM2019.

[22] K. Dorling, J. Heinrichs, G. M. Messier, and S. Magierowski, “Vehicle routing problems for drone delivery,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 1, pp. 70–85, 2017.

[23] J. Shi, P. Cong, L. Zhao, X. Wang, S. Wan, and M. Guizani, “A two-stage strategy for uav-enabled wireless power transfer in unknown environments,” IEEE Transactions on Mobile Computing, pp. 1–15, 2023.

[24] H. Huang and A. V. Savkin, “Towards the internet of flying robots: A survey,” Sensors, vol. 18, no. 11, 2018.

[25] A. Thibbotuwawga, P. Nielsen, B. Zbigniew, and G. Bocewicz, “Energy consumption in unmanned aerial vehicles: A review of energy consumption models and their relation to the uav routing,” in International Conference on Information Systems Architecture and Technology, pp. 173–184, Springer, 2018.

[26] I. Khouri, A. Lauiti, and C. Adjih, “A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles,” Drones, vol. 3, no. 3, 2019.

[27] W. Xu, Z. Xu, J. Peng, W. Liang, T. Liu, X. Jia, and S. K. Das, “Approximation algorithms for the team orienteering problem,” in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, pp. 1389–1398, 2020.

[28] 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. on Mobile Computing, pp. 1–1, 2021.

[29] Z. Liu, A. Kurzhanskiy, and R. Sengupta, “An energy-based optimal control problem for unmanned aircraft systems flight planning,” in 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 1320–1325, IEEE, 2017.

[30] DJI. DJI Matrice 100 User Manual, Mar 2016.

[31] T. A. Rodrigues, J. Patrikar, N. L. Oliveira, H. S. Matthews, S. Scherer, and C. Samaras, “Drone flight data reveal energy and greenhouse gas emissions savings for very small package delivery,” Patterns, vol. 3, no. 8, pp. 100569, 2022.

[32] Texas Instruments, LMT84 1.55-V, SC70/TQ-92/TQ-92S, Analog Temperature Sensors, Oct 2017.

[33] C. E. Miller, A. W. Tucker, and R. A. Zemlin, “Integer programming formulation of traveling salesman problems,” Journal of the ACM (JACM), vol. 7, no. 4, pp. 326–329, 1960.
[34] Texas Instruments, *HB Supercapacitors cylindrical cells (Technical Data 4375)*, Oct 2017.

[35] Amphenol Advanced Sensors, *NovaSensor NPA Surface Mount Pressure Sensors Datasheet*, 2019.

[36] Cooper Bussman, *PowerStor: Supercapacitors PHB Series Technical Sheet 4402*, 2011.

[37] L. Perron and V. Furnon, *Google OR-Tools*, 2023. v9.6. Available online: https://developers.google.com/optimization/.

[38] A. Hatamlou, “Solving travelling salesman problem using black hole algorithm,” *Soft Computing*, vol. 22, no. 24, pp. 8167–8175, 2018.

[39] G. Kobeaga, M. Merino, and J. A. Lozano, “An efficient evolutionary algorithm for the orienteering problem,” *Computers & Operations Research*, vol. 90, pp. 42–59, 2018.

[40] DJI, *DJI Manifold 2 Series User Guide*, May 2019.

[41] X. T. P. She, X. Lin, and H. Lang, “A data-driven power consumption model for electric uavs,” in *2020 American Control Conference (ACC)*, pp. 4957–4962, 2020.