A rapid unmanned aerial vehicle inspection path planning method based on hybrid heuristic algorithm

Boya Li, Chenxi Wu*, Jian Ding, and Difeng Zhu
State Grid Zhejiang Electric Power CO., LTD. UHV Branch, China

Abstract. The advancement of unmanned aerial vehicle (UAV) and remote sensing technologies has fueled interest in automatic UAV inspection path planning based on inspection tasks. However, traditional methods suffer from limitations such as manual operation, inability to find optimized paths, and lengthy time consumption. We propose a rapid UAV inspection path planning method using a hybrid heuristic algorithm in this paper. First, real-world data is abstracted into a graph. Then, a hybrid heuristic algorithm is proposed and used to determine an optimal inspection path considering the tasks and power parameters of the UAV. Finally, the total power consumption for the mission is computed to assess UAV battery capacity adequacy. Experimental results demonstrate the effectiveness of our method in terms of both performance and accuracy.

1 Introduction

As an inseparable part of the power grid, the secure and stable operation of substations is of utmost significance, thus regular status inspections of substations are required [1]. With the rapid evolution of unmanned aerial vehicle (UAV) and remote sensing technologies, UAVs have emerged as a focal point for power transmission line inspections due to their lightweight, compact size, cost-effectiveness, and adaptability [2]. Moreover, integrating inspection sensors into UAV systems has made them an efficient and cost-effective method for substation inspections [2].

Traditional inspection methods often require manual or semi-automatic operation for path planning and UAV control [3], posing challenges in path planning and UAV operations. Employing intelligent algorithms for UAV path planning and optimization can significantly reduce manual labor and inspection time. By abstracting UAV inspection tasks into the traveling salesman problem (TSP), various metaheuristic algorithms, such as simulated annealing (SA), ant colony optimization, bees algorithm (BA), and genetic algorithm (GA), have been proposed and applied to handle TSP [5].


Despite advancements in deep learning technologies for TSP, their widespread application faces challenges due to slow computations and high hardware requirements [11]. Balancing computational capability requirements and speed is crucial for real-time planning and optimizing substation inspection paths. In this paper, we propose a rapid UAV inspection path planning method based on a hybrid heuristic algorithm to enhance optimal solution discovery and reduce time overhead. Additionally, we integrate the algorithm into a UAV control app for automated inspection path planning.

2 Proposed method

2.1 Method overview

The framework of the proposed method, as illustrated in figure 1, consists of three main modules: abstract graph creation, inspection path planning, and total power consumption calculation.

Fig. 1. Overall framework diagram.

Firstly, inspection targets, channel information, UAV tasks, and power consumption parameters are abstracted into a graph with nodes and edges. Then, a hybrid heuristic algorithm is utilized to plan the optimal inspection path considering UAV tasks. Finally, the total power consumption for the current task is computed based on the planned path, and the feasibility of completing the task with the current UAV battery level is assessed.

2.2 Creation of the abstract graph

For UAV inspection path planning, the method abstracts actual inspection devices and channel information into a graph. This involves converting device locations into nodes and channel information into edges. We assign weights to nodes and edges based on UAV tasks (such as taking photos, filming) and power consumption parameters (such as level flight, hover, lift). The algorithm is shown in figure 2.
The power consumption parameters are detailed in Table 1, with specific tasks and their values depending on the specific program.

**Table 1. Task power consumption parameter unit.**

<table>
<thead>
<tr>
<th>Task</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level Flight</td>
<td>mA·h/s</td>
</tr>
<tr>
<td>Hover</td>
<td>mA·h/s</td>
</tr>
<tr>
<td>Lift</td>
<td>mA·h/m</td>
</tr>
<tr>
<td>Taking Photos</td>
<td>mA·h/sheet</td>
</tr>
<tr>
<td>Filming</td>
<td>mA·h/s</td>
</tr>
</tbody>
</table>

We calculate the power consumption of each task at the target location and add it to the node weights as shown in equation (1).

$$e_{c_{node}} = \sum_{i=1}^{N} (task_{time}^i \times task_{consumption}^i)$$  \hspace{1cm} (1)

where $e_{c_{node}}$ represents the power consumption weight of the node, $N$ denotes the total number of tasks at the node, $task_{time}^i$ stands for the time needed to execute the $i^{th}$ task, and $task_{consumption}^i$ represents the power consumption parameters of the $i^{th}$ task.

Using the power consumption parameters and channel information, the power consumption of each edge is calculated and the edge weights are updated accordingly as shown in equation 2.

$$e_{c_{edge}} = \frac{len_{edge}}{v} \times consumption_{fly} + time_{hover} \times consumption_{hover}$$  \hspace{1cm} (2)

where $e_{c_{edge}}$ represents the edge's power consumption weight, $len_{edge}$ is the edge length, $v$ is the flight speed of the UAV, $consumption_{fly}$ and $consumption_{hover}$ correspond to the power consumption parameters during flighting and hovering, respectively, and $time_{hover}$ is the UAV’s hovering time.

### 2.3 Inspection path planning

The UAV’s battery capacity is fixed, meaning that minimizing power consumption during inspection tasks allows for a greater number of tasks to be performed. To achieve this goal, we propose a rapid hybrid heuristic algorithm for UAV path planning during inspections. This algorithm comprises three sub-algorithms: enumeration, improved genetic simulated annealing algorithm (IGSAA), and improved combinatorial bees algorithms (ICBA).
2.3.1 Enumeration algorithm

All possible inspection paths are obtained by enumeration algorithm and the power consumption of each path is calculated, as shown in equation 3.

\[ ec^i = \sum_j ec^{i,j}_{\text{node}} + \sum_k ec^{i,k}_{\text{edge}} \]  

(3)

where \( ec^i \) represents the power consumption of the \( i-th \) enumerated path, \( j \) denotes the number of nodes in the path, \( ec^{i,j}_{\text{node}} \) is the power consumption at the \( j-th \) node of the \( i-th \) path, \( K \) is the number of edges in the path, and \( ec^{i,k}_{\text{edge}} \) is the power consumption of the \( i-th \) edge of the \( k-th \) path.

Next, the inspection path with the lowest power consumption as the optimal path, as shown in equation 4.

\[ path_{\text{enum}}^{\text{best}} = \min_{1 \leq i \leq N_{\text{enum}}} ec^i \]  

(4)

where \( path_{\text{enum}}^{\text{best}} \) represents the optimal path for the enumeration algorithm, and \( N_{\text{enum}} \) is the total number of enumerated paths.

2.3.2 Improved genetic simulated annealing algorithm

Traditional genetic algorithms often struggle in late optimization stages when population individuals possess highly similar chromosomes, hindering the generation of new chromosomes post-crossover and impeding escape from local optima. On the other hand, simulated annealing algorithms have a strong crawling ability and do not easily fall into a local optimum, but mutations may produce worse solutions. Therefore, we integrate genetic algorithm with simulated annealing algorithm, optimizing their operators and mutation processes to form an improved genetic simulated annealing algorithm.

1) Initializing Populations and Calculating Fitness

According to the total number of UAV inspection targets \( num_{\text{target}} \), \( m \) initial populations of length \( num_{\text{target}} + 1 \) are generated. Calculate the power consumption of each path in the initial population and assign the minimum power consumption and its path to the local best power consumption \( ec_{\text{local}}^{\text{best}} \), the global best power consumption \( ec_{\text{global}}^{\text{best}} \), the local best path \( U_{\text{local}}^{\text{best}} \) and the global best path \( U_{\text{global}}^{\text{best}} \), respectively. Initial temperature \( T_0 \), current temperature \( T \), and minimum temperature \( T_{\text{min}} \) are defined to regulate the annealing process, with \( T_0 = 100, T_{\text{min}} = 0.1 \). Additionally, the maximum number of iterations \( g_{\text{max}} \) at each temperature is defined, with the current iteration number \( g \) initialized to 1.

To maintain population diversity and enhance the algorithm's ability to avoid local optimums, a fitness function is introduced so that individual population differences become smaller with decreasing temperatures. Therefore, individuals with lower fitness are more likely selected at lower temperatures, as shown in equation 5.

\[ f(ec) = \begin{cases} (ec_{\text{global}}^{\text{best}} - ec) \left( \frac{T}{100} \right)^{\frac{3}{2}}, & ec < ec_{\text{max}} \\ 0, & ec \geq ec_{\text{max}} \end{cases} \]  

(5)

where \( f(ec) \) denotes the fitness function, and \( ec_{\text{max}} \) represents the maximum allowable power consumption.

2) Improved Genetic Algorithm and Jump Change

The improved genetic algorithm in this paper optimizes the genetic operators as follows:
Selection Operator: Based on the fitness values of the individuals, \( m \) repeatable individuals were randomly selected from the population and \( n \) of them were randomly replaced.

Crossover Operator: Selected individuals were paired and randomly crossed \( nc \) genes with a probability of \( P_c \). Individuals were adjusted after crossover to eliminate duplicate genes.

Mutation Operator: The chromosomes of individuals after crossover are mutated with probability \( P_b \) to randomly shift or swap their genes.

Calculate the power consumption of each individual after genetic, and modify each individual according to the following rules:

(a) If the power consumption decreases after genetic, retain the individual in the new population.

(b) If the power consumption increases after genetic, the genes are exchanged with other genes with probability \( P_1 \) with a probability of 0.02 for each gene. The jump change probability \( P_1 \) is shown in equation 6.

\[
P_1 = \exp \left(-\frac{u_1(e_{after} - e_{before})}{T} \right)
\]

where \( u_1 \) is the decay coefficient of probability \( P_1 \), which takes the value of 3000, \( e_{before} \) and \( e_{after} \) are the power consumption before and after the genetic algorithm.

(c) If the power consumption remains unchanged after genetic, two random sets of consecutive genes are exchanged with probability \( P_2 \) randomly with probability 0.1, and the crossover remains unchanged. The jump change probability \( P_2 \) is shown in equation 7.

\[
P_2 = \exp \left(-\frac{u_2}{T} \right)
\]

where \( u_2 \) is the decay coefficient of the probability \( P_2 \), which takes the value of 3.

3) Update and Iteration

After jump change, power consumption for each individual in the new population is computed. The local best power consumption \( e_{locbest} \) and path \( U_{locbest} \) are updated by comparing with the current minimum power consumption. If \( e_{locbest} < e_{globest} \), the global best power consumption \( e_{globest} \) and path \( U_{globest} \) are updated.

After reaching the iteration limit \( g_{max} \), \( T \) is updated for annealing, as shown in equation 8.

\[
T = 0.9T
\]

The loop terminates when \( T < T_{min} \), obtaining \( e_{globest} \) and \( U_{globest} \), otherwise, iteration persists.

2.3.3 Improved combinatorial bees algorithm

Due to the randomness of initial populations, conventional bees algorithms often encounter difficulties in identifying global optimal solutions. Thus, an improved combinatorial bees algorithm is proposed, refining both initial population generation and search algorithms.

1) Population initialization

The population size \( n \) is set based on the total number of inspection targets \( nc \) for the UAV, with an equal number of scout bees. Parameters like local search honey sources \( m \), elite honey sources \( e \), scout bees recruited by elite honey sources \( nep \), scout bees recruited by other local honey sources \( nsp \), neighborhood size \( ngh \), and iteration count \( l \) are also defined.

The initial population is generated using NNM, where each location is sequentially added along with its nearest neighbors until the population size is reached. Although NNM incurs some overhead, it's acceptable as it's only executed once in the whole algorithm.
2) Search algorithm

The search algorithm contains global and local search sections. In the global search section, random paths are less effective and the use of NNM increases the computational requirement, making it not suitable for real-time path planning, so the global search section is not used in the proposed ICBA. In the local search section, the algorithm includes insert, multi-insert, double member swap, and reversion operators, as shown in figure 3.

(a) Insert Operator
During the insert operation, a location is randomly selected and inserted into a different randomly selected order. In the paths created using NNM, there are few connections between distant locations. Therefore, to improve the path, the method selects neighbors in the local search part. However, selecting only the nearest neighbors may limit the variation and may lead to finding only local optima, so neighbors are defined as the nearest 5-10 locations.

(b) Multi-Insert Operator
The multi-insert operator selects 2 to 5 locations from a randomly chosen order and inserts them to a second random order within the neighborhood. This operator is implemented in two different ways. In the first way the selected locations are inserted into the other areas in the same order, while in the second way the order of the locations is reversed.

(c) Double Member Swap Operator
In the double member swap operator, the positions of two randomly determined locations are swapped, which is effective in solving the problem of intersecting edges that leads to an increase in the total length.

(d) Reversion Operator
The reversion operation reverses the order between two randomly determined locations, which is particularly effective in correcting path sections with edge crossings and consequently reducing the total path length.

Fig. 3. Search operators, a, b, c, d for insert, multi-insert, double member swap and reversion.

2.3.4 Algorithm selection

For each detection task, the choice of algorithm depends on the number of targets. When the number of targets is less than 10, the enumeration algorithm provides the global optimal solution, while when the number of targets is more than 10, the heuristic algorithm is preferred due to the time constraints. In order to prevent the heuristic algorithm from falling
into local optimum, two heuristic algorithms are used and the one with lower power consumption is chosen.

3 Experiment

To validate the effectiveness and performance of our method, we conducted two sets of experiments: one using simulated environment data and the other using real-world data.

3.1 Experiments in simulated environment

In the simulated environment experiments, we generated a randomized abstract graph comprising 690 inspection devices and 788 channels, as shown in figure 4. This environment remained constant throughout the experiments for controlled comparison. The virtual environment used in this experiment was an x86-based Android 11.0 Google Play image.

![Randomly generated abstract graph.](image)

We varied the number of inspection targets, selecting 8, 50, and 100 devices to assess algorithm performance across different scales. Each set of targets formed the initial inspection path, tested 10 times to mitigate randomness. Average execution time and power consumption were recorded, as shown in table 2.

<table>
<thead>
<tr>
<th>Number of inspection equipment(unit)</th>
<th>Method</th>
<th>Average power consumption(mA·h)</th>
<th>Average execution time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Enumeration algorithm</td>
<td>1677.61</td>
<td>0.239</td>
</tr>
<tr>
<td></td>
<td>Genetic algorithm</td>
<td>1831.51</td>
<td>1.812</td>
</tr>
<tr>
<td></td>
<td>Our Method</td>
<td>1677.61</td>
<td>0.163</td>
</tr>
<tr>
<td>50</td>
<td>Enumeration algorithm</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Genetic algorithm</td>
<td>12758.53</td>
<td>15.062</td>
</tr>
<tr>
<td></td>
<td>Our Method</td>
<td>3418.33</td>
<td>14.741</td>
</tr>
<tr>
<td>100</td>
<td>Enumeration algorithm</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Genetic algorithm</td>
<td>22765.52</td>
<td>41.647</td>
</tr>
<tr>
<td></td>
<td>Our Method</td>
<td>4664.24</td>
<td>28.720</td>
</tr>
</tbody>
</table>

For inspection device counts exceeding 10, the enumeration method's time cost surpassed acceptability, left unreported. Compared to the enumeration algorithm, our method overcomes the computational limitations, has a wider range of applications, and is more...
efficient. In addition, it outperforms the genetic algorithm in terms of path quality and computational speed.

### 3.2 Experiments in real-world environment

In the real-world environment, the UAV needs to conduct inspection tasks at substations, as shown in figure 5. For the experiments, we selected a local area of the substation, and its abstract graph is shown in figure 6.

![Fig. 5. Actual scenario picture.](image1)

![Fig. 6. Actual scenario abstract graph.](image2)

Due to battery limitations, inspection targets per task typically stay under 15. Thus, we selected 5 and 11 devices for UAV inspection. The experimental results are shown in table 3. The actual machine result is shown in figure 7, and the device is the DJI Royal 3Pro industry machine controller.

<table>
<thead>
<tr>
<th>Number of inspection equipment(unit)</th>
<th>Takeoff point</th>
<th>Average power consumption(mA·h)</th>
<th>Average execution time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>2219.84</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2615.33</td>
<td>0.007</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1692.40</td>
<td>0.944</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2130.77</td>
<td>0.957</td>
</tr>
</tbody>
</table>
4 Conclusion

We propose a rapid UAV inspection path planning method based on a hybrid heuristic algorithm to minimize power consumption in the paper, which comprises three modules: abstract graph creation, path planning, and total power consumption calculation. Firstly, location and channel information are abstracted into nodes and edges, with weights determined based on task and UAV parameters to construct the abstract graph. Secondly, the enumeration or heuristic algorithm is selected based on the number of inspection targets to calculate the optimal path. Finally, the total power consumption is computed based on the optimal path and current UAV battery level to determine task feasibility. Experiments have shown the method has excellent performance and accuracy. In the future, we plan to investigate more heuristic algorithms to further improve the accuracy and performance of the calculation results.

This work was partially supported by the Science and Technology Project of State Grid Zhejiang Electric Power CO., LTD. (No. 5211MR230002).

References


