2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Heuristic Navigation Model based on Genetic Programming for Multi-UAV Power Inspection Problem with Charging Stations

Xiang-Ling Chen, Xiao-Cheng Liao, and Wei-Neng Chen, *Senior Member, IEEE*

***Abstract*— Efficient power inspection is crucial for maintaining a stable power system. During an inspection, unmanned aerial vehicles (UAVs) usually need to be recharged due to the wide geographical range of inspection and the limited battery capacity of UAVs. This limitation makes the problem more challenging that requires not only optimizing the task execution order, but also taking the chargings of UAVs into consideration. In order to address this complex problem, this work first formulates the UAV power inspection planning problem with charging stations. After that, we propose a new heuristic navigation model, in which UAVs can follow a heuristic rule to decide where to go next based on both its own information and task-related information. To obtain the heuristic rule, we design a set of features to describe the status of the UAVs and task completion. Then a genetic programming (GP) algorithm is introduced to evolve and get the heuristic rule. Finally, by applying heuristic navigation rule, the UAV navigation model can automatically prioritize task and charging order, and generate UAV flight routes that satisfy all constraints. The experiment results show that our method significantly outperforms the state-of-the-art algorithms.**

***Index Terms*— unmanned aerial vehicles (UAVs), task assignment problem, charging problem, heuristic navigation model, genetic programming**

I. INTRODUCTION

Power system is an essential infrastructure for modern society. Electricity is transmitted through power lines. The efficient monitoring of equipment such as power poles, as well as the timely detection and localization of faults, have emerged as critical issues that the power industry must address urgently. Compared to traditional inspection methods that are labor-intensive and inefficient, the use of unmanned aerial vehicles (UAVs) [1], [2] in power line inspections has the potential to reduce costs, increase efficiency, and enhance safety by minimizing the need for human intervention, especially in hard-to-reach or accident-prone power inspection regions. Therefore, UAV power inspection technology has attracted more and more attention from the power industry [3], [4].

Power grids are commonly established in mountainous regions, where the area requiring inspection for power operations is often extensive, the current flight distance of UAVs is inadequate to encompass the inspection area without necessitating recharging. Therefore, in order to ensure that all power poles in the huge inspection area are inspected successfully, it is necessary to consider recharging the UAVs [5]. If the UAV fails to complete all of its assigned tasks with its initial energy reserves, it can go to the charging stations halfway to recharge.

For power inspection problem, it can be modeled as a task assignment problem, with each power pole representing a task that should be inspected once. After inspecting all tasks, all UAVs should go back to the depots or charging stations. Many forms of task assignment problems have been proven to be NP-hard [6] that they are very challenging to find an optimal solution in polynomial time. The existence of charging stations makes it more difficult to manage the UAVs, as we must consider two aspects: (1) all tasks must be completed effectively and (2) the UAV must obey the constraints of battery power availability during flight route. These two aspects are interrelated and further complicate the whole problem.

So far, several optimization approaches have been proposed to solve the task assignment problem with charging stations (TAPCS) in the literature. They can be generally classified into two categories: 1) centralized methods; 2) bi-level optimization methods. Centralized methods are to consider both task assignment and recharging scheduling in the optimization process. They try to solve the whole problem simultaneously by encoding the task order and charging station together [7], [8]. They can generate very good solutions on small-scale problems, but the huge search space produced by the encoding scheme makes them inefficient on large-scale problems. The bi-level optimization methods divide the TAPCS problem into two levels, with the upper level utilizing meta-heuristic algorithms [9]–[11] to optimize the order of task assignment and the lower level optimizing the fixed routes UAV charging problem [12], [13]. In lower level problem, using the enumerated method that enumerates all possible charging station plug-in locations can work well if the flight route requires only a few charging times, but it becomes disastrous when multiple charging times are required and there is an abundance of optional charging stations [14]. Applying meta-heuristic algorithms [15]–[17] in both two levels also leads to extremely high time complexity and computational expense. Therefore, some efficient methods, such as heuristic algorithms, are usually adapted to address the lower level problem in an accepted execution time [12], [18]. They can effectively solve the fixed route charging problem. However, human-designed heuristic charging algorithms do not take the information of the UAV

This work was supported in part by the National Natural Science Foundation of China under Grants 61976093 and Guangdong Regional Joint Foundation Key Project 2022B1515120076. *(Corresponding Author: Wei-Neng Chen, cschenwn@scut.edu.cn)*

Xiang-Ling Chen, Xiao-Cheng Liao and Wei-Neng Chen are with the School of Computer Science and Engineering and with the State Key Laboratory of Subtropical Building and Urban Science, South China University of Technology, Guangzhou 510006, China. *(Email: Wei-Neng Chen, cschenwn@scut.edu.cn)*

2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3503-3702-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMC53992.2023.10394169

itself into account, which may lead to unreasonable selection of the location of charging station, incurring additional charging costs [12].

Generally speaking, it is very challenging to design effective heuristic rules manually, as the selection rule has to consider various factors like the remaining power of UAVs, the distances to the next task, the spatial distribution of the remaining tasks, etc. Thus a flexible and intelligent scheduling method is required to decide whether to charge and where to charge. Genetic programming (GP) can automatically learn scheduling heuristics without human intervention [19]–[21]. The design of heuristic rules is associated with the UAV's own status, tasks, and charging station location information. Therefore, it may be more flexible and adaptable to deal with the TAPCS problem.

Considering these difficulties of the problem and the drawbacks of the existing studies, we propose a novel UAV heuristic navigation model based on a heuristic rule evolved by genetic programming (GP). The main contributions are:

1) We propose a novel UAV heuristic navigation model to solve the TAPCS problem. Based on a heuristic rule, the proposed model can simulate the process of UAV flight to generate a feasible solution for a given TAPCS problem instance. It can automatically guide the activities of UAVs, so that the UAV can automatically decide where to go next based on its own information and surrounding environment information.
2) A unique set of both UAV-related and task-related features is designed and serves as the variables of the heuristic rule.
3) We adapt GP to find the optimal heuristic rule to address the TAPCS problem and the experimental results demonstrate our method can outperform the other compared methods.

The rest of this paper is organized as follows. Section II formulates the multi-UAV power inspection problem with charging stations. The proposed algorithm is explained in Section III. The experimental results are analyzed in Section IV. Finally, the conclusions are drawn in Section V.

II. Problem Definition

In the multi-UAV power inspection problem with charging stations, there are M power poles to be inspected, and each pole can be viewed as a task t_i to be done, i.e., $T = \{t_1, t_2, ..., t_M\}$. Simultaneously, there are NC charging stations $C = \{c_1, c_2, ..., c_{NC}\}$. There is a set of UAVs $U = \{u_1, u_2, ..., u_l\}$ to complete these tasks. Each UAV has a maximum battery capacity Q and a fixed flight speed v. The battery's consumption rate is denoted as h. UAVs take off from a charging station c_1 and can eventually return to any charging station [22]. TAPCS can be modeled as a undirected graph $G = (V, E)$. $V = T \cup C$, $E = \{(i, j) \mid i, j \in V, i \neq j\}$ is the set of edges. Each edge $e_{ij} = (v_i, v_j)$ is associated with a distance d_{ij} from v_i to v_j. UAVs will consume $h \cdot d_{ij}$ battery power for edge e_{ij}. Each task requires a fixed completion time τ, and the recharging time is negligible. q_{u_k} represents the current battery power of the UAV u_k.

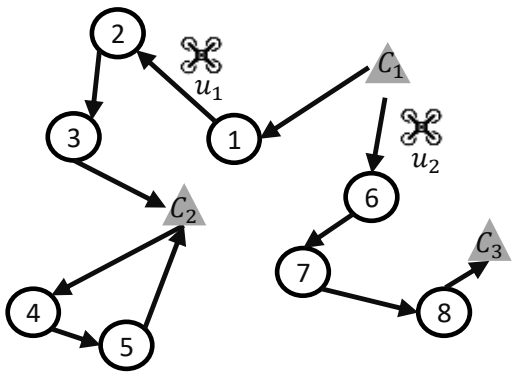


Fig. 1. Example of a solution of a TAPCS problem.

Given a problem instance, the problem is to design an execution plan for UAVs that they need to complete all of tasks while adhering to the constraints. It can be defined as follows:

$$min\ f(x) = \omega_1 \cdot dis(x) + \omega_2 \cdot c(x) + \omega_3 \cdot ct(x) \tag{1}$$

s.t.

$$\sum_{i \in V, i \neq j} x_{ij} = 1,\ \forall j \in T \tag{2}$$

$$end(u_i) \in C \quad \forall u_i \in U \tag{3}$$

$$0 \leq q_{u_k} \leq Q\ \forall j \in x_{ij}^{u_k} \tag{4}$$

$$x_{ij} = \{0, 1\}, i \neq j\ \forall i, j \in V \tag{5}$$

The objective (1) is to minimize the weighted sum of the UAVs' total flight distance $dis(\cdot)$, total recharging times $c(\cdot)$ and the completion time of the last task $ct(\cdot)$, where ω_1, ω_2, ω_3 are the coefficients which represent the relative of the three costs. Constraint (2) restricts that every task should be completed only once. Constraint (3) guarantees that every UAV must return to the charging station. Constraint (4) restricts the UAV must ensure that the battery power is feasible during the journey. Constraint (5) defines the domain of x_{ij}. If there is a path for UAV to go from v_i to v_j, x_{ij} takes 1, otherwise it equals to 0.

The calculation formulas of $dis(\cdot)$, $c(\cdot)$ and $ct(\cdot)$ are as follows:

$$dis(x) = \sum_{i,j \in V, i \neq j} d_{ij} x_{ij} \tag{6}$$

$$c(x) = \sum_{i \in V, j \in C, i \neq j} x_{ij} \tag{7}$$

$$ct(x) = \max_{u_k \in U} \{dis(x^{u_k})/v + \tau \sum_{i \in V, j \in T, i \neq j} x_{ij}^{u_k}\} \tag{8}$$

To better understand the TAPCS problem, an example of a solution on an instance with two UAVs is shown in Fig. 1. The UAVs all start from c_1 to complete tasks. u_1 visited the charging station c_2 for recharging during its flight and return to c_2, and another UAV does not recharging and return to c_3.

III. Proposed Method

In this section, we present the proposed heuristic navigation model for UAVs that employs a rule $\Gamma(\cdot)$ to generate feasible solutions for TAPCS problem. Furthermore, we provide an overview of the representation of the rule $\Gamma(\cdot)$. Finally, we describe the evolutionary process of GP that is utilized to evolve $\Gamma(\cdot)$ for TAPCS problem.

A. UAV heuristic navigation model

Given a TAPCS instance $\mathcal{H}$ and a rule $\Gamma(\cdot)$, the UAV heuristic navigation model can generate a solution by applying $\Gamma(\cdot)$ to the instance. The solution can be represented as a set of UAV flight routes:

$$\mathcal{S} = \{F_1, F_2, ...F_l\} \tag{9}$$

Every flight route is represented as a permutation of tasks and charging stations, that is:

$$F_p = [c_1, t_p^1, c_p^1, t_p^2, ..., c_p^i, t_p^m, c] \tag{10}$$

The proposed UAV heuristic navigation model is presented in Algorithm 1. At the initialization stage (lines 1-7), the solution $\mathcal{S}$ is set to empty. Since none of the tasks are completed, we set available nodes $\mathcal{A} \leftarrow T \cup C$. All UAVs are at charging station c_1, we therefore initialize the flight route of each UAV as $F_k \leftarrow \{c_1\}$, and the current positions are set as $P_{u_k} \leftarrow c_1$. The active UAVs set AU is initialized to U.

At the start of the simulation, the next steps for each active UAV are determined during the decision-making stage (lines 10-38). The decision-making stage can be decomposed into two phases: 1) selection phase: select the highest scoring node for the UAV and 2) UAV movement phase: UAV moves to the selected node if the node exists.

In the selection phase, for each active UAV, we first remove the nodes from the available nodes $\mathcal{A}$ that the UAV cannot visit. The removed nodes include:

1) Remaining battery power of UAVs is insufficient to reach the node v_i (lines 11-13).
2) The UAV is currently at the charging station, and the node v_i is also a charging station (lines 14-16). The limitation can prevent unnecessary shuttle trips between charging stations because we assume that UAVs are always fully charged, and traveling from charging station to charging station means more additional time and distance costs.

Then, we initialize the maximum priority value max_p as $-\infty$ and the next node to visit $next\_v$ as empty (lines 17-18). UAV calculates the priorities of the remaining available nodes according to the given rule $\Gamma(\cdot)$ and selects the node with the highest priority to move next (lines 19-23).

During UAV movement phase, if there is no $next\_v$ in the upper selection phase, it means that all tasks are completed and the UAV simulation ends. Therefore, we remove this UAV from the active UAVs AU (line 25-27). Otherwise, if $next\_v$ is a task node, we remove it from the available nodes $\mathcal{A}$ because the task can only be visited once and update the UAV's battery power (lines 28-31). If $next\_v$ is a charging station node, we only update the UAV's battery power to the maximum battery capacity Q (line 32). Since the charging stations can be visited multiple times, C always belongs to $\mathcal{A}$. Finally, we add $next\_v$ to the flight route F_k and update the UAV u_k's current position (lines 34-35).

The decision-making stage will run repeatedly until there are no active UAVs. After the simulation is finished, the solution $\mathcal{S}$ can be obtained by merging each UAV's flight routes F_k (line 39).

Algorithm 1: UAV heuristic navigation model

Input : A TAPCS instance $\mathcal{H}$, a rule $\Gamma(\cdot)$
Output: solution $\mathcal{S}$

1 $\mathcal{S} \leftarrow \emptyset$
2 set available nodes $\mathcal{A} \leftarrow T \cup C$
3 **for** *each* $u_k \in U$ **do**
4 $\quad F_k \leftarrow [c_1]$
5 $\quad P_{u_k} \leftarrow c_1$
6 **end**
7 set active UAVs $AU \leftarrow U$
8 **while** *AU is not empty* **do**
9 $\quad$ **for** $u_k \in AU$ **do**
10 $\quad\quad max\_p \leftarrow -\infty$
11 $\quad\quad next\_v \leftarrow \emptyset$
12 $\quad\quad$ **for** $v_i \in \mathcal{A}$ **do**
13 $\quad\quad\quad$ **if** $q_{u_k} \leq h \cdot d_{i,P_{u_k}}$ **then**
14 $\quad\quad\quad\quad$ continue
15 $\quad\quad\quad$ **end**
16 $\quad\quad\quad$ **if** $v_i \in C$ *and* $P_{u_k} \in C$ **then**
17 $\quad\quad\quad\quad$ continue
18 $\quad\quad\quad$ **end**
19 $\quad\quad\quad$ Calculate the priority value p_i of nodes according to $\Gamma(\cdot)$
20 $\quad\quad\quad$ **if** $max\_p < p_i$ **then**
21 $\quad\quad\quad\quad max\_p \leftarrow p_i$
22 $\quad\quad\quad\quad next\_v \leftarrow v_i$
23 $\quad\quad\quad$ **end**
24 $\quad\quad$ **end**
25 $\quad\quad$ **if** $next\_v$ *is* $\emptyset$ **then**
26 $\quad\quad\quad AU \leftarrow AU \setminus \{u_k\}$
27 $\quad\quad$ **else**
28 $\quad\quad\quad$ **if** $next\_v \in T$ **then**
29 $\quad\quad\quad\quad \mathcal{A} \leftarrow \mathcal{A} \setminus \{next\_v\}$
30 $\quad\quad\quad\quad q_{u_k} \leftarrow q_{u_k} - h \cdot d_{P_{u_k}, next\_v}$
31 $\quad\quad\quad$ **else**
32 $\quad\quad\quad\quad q_{u_k} \leftarrow Q$
33 $\quad\quad\quad$ **end**
34 $\quad\quad\quad F_k \leftarrow F_k \cup [next\_v]$
35 $\quad\quad\quad P_{u_k} \leftarrow next\_v$
36 $\quad\quad$ **end**
37 $\quad$ **end**
38 **end**
39 $\mathcal{S} = \{F_k | \ \forall \ u_k \in U\}$
40 **return** $\mathcal{S}$

TABLE I
TERMINAL SET FOR THE GP ALGORITHM

Symbol	Description
TN	Type of the candidate Node
DFH	Distance From Here (UAV position) to the candidate node
DFNT	Distance From the candidate Node to its closest remaining Task
DFNC	Distance From the candidate Node to its closest Charging station
RE	Remaining battery power of the UAV
FRT	Fraction of Remaining Tasks
Constant	ephemeral terminal [23] from -10 to 10

B. $\Gamma(\cdot)$ Representation

In the proposed GP method, each $\Gamma(\cdot)$ is represented as a tree-based priority function that can output the most suitable node $v_i \in V$ for UAV to move forward. When the UAV need to decide the next activity, $\Gamma(\cdot)$ is used to calculate the priority of each candidate node. A $\Gamma(\cdot)$ consists of functions and terminals. In this work, there are 6 functions in function set:

$$\mathbb{F} = \{+, -, \times, \div, min, max\} \tag{11}$$

where "$\div$" is the protected division that returns 1 if divided by zero.

To build the heuristic rule $\Gamma(\cdot)$, we analyze and extract the UAV-related and task-related features such as the UAV's remaining battery power, location of UAV, tasks and charging stations. Then we combine these features into 6 variables to design the terminal set. The detail is given in Table I, where symbol is the name of the variable and on the right is the description. TN represents the type of the candidate node, it equals to 1 if the node belongs to T, otherwise it equals to 0. Additionally, a special terminal named "*ephemeral random constant*" [23] is also utilized in the terminal set.

Fig. 2 shows the tree representation of an example $\Gamma(\cdot)$. This tree can be converted into a mathematical formula:

$$\Gamma(\cdot) = TN - min(DFH, DFNT) + RE - DFH \tag{12}$$

According to (12), the priority value of each candidate node v_i can be calculated. Therefore, UAVs are able to decide which node to go to (i.e., task or charging station) based on the priority values.

C. Evolutionary Process

Gene expression programming (GEP) [24] is a famous variant of GP, which encodes in linear chromosomes of fixed length. The structural organization of linear chromosomes allows for unrestricted manipulation and ensures that the resulting chromosomes are valid. In this work, we employ GEP to evolve the optimal $\Gamma(\cdot)$.

1) Fitness function representation:

Because the TAPCS problem is a combinatorial optimization problem with constraints, a penalty function is added to the objective function to penalize the infeasible solutions [25].

$$\mathcal{G}(\mathcal{S}) = f(\mathcal{S}) + (UT + H) \times 10^4 \tag{13}$$

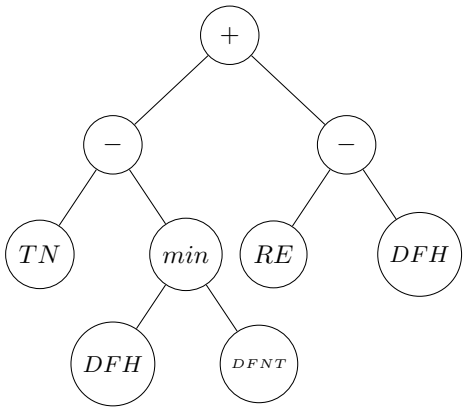


Fig. 2. Example of a rule of a TAPCS problem.

where UT is the number of unfinished tasks, H is the number of UAVs that have not returned to charging station. If an infeasible solution is obtained by using $\Gamma(\cdot)$, that is, UT or H is not 0, that a higher value $(UT+H)\times 10^4$ will be penalized into the fitness value. As a result, the infeasible solutions can be compared to one another, and their fitness values will be larger than those of all feasible solutions.

2) Genetic operators:

Selection: The selection operator select the individual form the population by tournament selection with elitism.

Mutation: There are four mutation operators including 1) point mutation, 2) inversion mutation 3) insertion sequence (IS) mutation and 4) root insertion sequence (RIS) mutation. All of these operations are standard operations present in [24].

Crossover: The one-point crossover operator and two-point crossover operator [24] are also employed in the proposed GP.

3) GP algorithm:

The overall framework of GP is presented in the Algorithm 2. The algorithm begins with the random generation of the initial population's chromosomes (line 1). Then the evolutionary process starts. Firstly, individuals are expressed and the fitness of each individual is calculated by (13) (lines 3-7). Then, the best individual is preserved for the next generation, while the remaining individuals are selected by the tournament selection operator based on fitness values. After that, individuals of this new generation go through the same developmental process: chromosomal expression, selection, mutation and crossover (lines 3-14). The process is repeated until the maximum number of generations is reached, and the best individual is returned as $\Gamma(\cdot)$.

D. Time Complexity

In UAV heuristic navigation model, the UAVs need to make decisions to calculate the priority of candidate nodes. Based on Table I, the time of calculating priorities is related to the number of the accessible node set $\mathcal{A}$. To complete all N tasks, at least N decisions are required. Each decision requires calculating the priorities of $|\mathcal{A}|$ nodes. Therefore, the time complexity is $O(N \cdot |\mathcal{A}|^2)$.

Algorithm 2: GP-based $\Gamma(\cdot)$ generation

Input : A TAPCS instance $\mathcal{H}$
Output: A near-optimal rule $\Gamma(\cdot)$

Create chromosomes and initial population P
while $iter \leq max\_Iter$ **do**
  for *each* $r \in P$ **do**
    Express r
    $\mathcal{S} \leftarrow$ UAV heuristic navigation model $(\mathcal{H}, r)$
    Calculate fitness value of $\mathcal{S}$ obtained from r by (13)
  end
  Keep the best r to the next generation
  $P \leftarrow$ Tournament selection (P)
  $P \leftarrow$ Point Mutation (P)
  $P \leftarrow$ IS Mutation (P)
  $P \leftarrow$ RIS Mutation (P)
  $P \leftarrow$ 1-point Crossover (P)
  $P \leftarrow$ 2-point Crossover (P)
  $iter \leftarrow iter + 1$
end
for *each* $r \in P$ **do**
  Express r
  $\mathcal{S} \leftarrow$ UAV heuristic navigation model $(\mathcal{H}, r)$
  Calculate fitness value of $\mathcal{S}$ obtained from r by (13)
end
$\Gamma(\cdot) \leftarrow$ best r from P
return $\Gamma(\cdot)$

IV. Experiments

In this section, we construct a series of instances to simulate TAPCS problems. Based on these instances, the proposed method is compared with some algorithms and the experimental results are analyzed in detail.

A. Experiment Settings

1) Instances:

To investigate the performance of GP, total 11 instances are constructed. The instances are different in the number of UAVs, tasks, and charging stations. The location of the charging stations are uniformly distributed in the inspection area. The UAVs all start from one of the charging stations and can eventually return to any charging station. The task locations are generated randomly in a 2-D plane. The specific design of each instance is shown in Table II.

2) Cost parameters:

In actually, the cost criteria ω_1, ω_2, ω_3, h and Q are indeed task-related and UAV-related and they should be decided by the decision maker, i.e., energy companies [26]. In this work, we take the setting of (5.0, 5.0, 20.0, 1, 20) for an example.

3) GEP setting:

We set the population size NP = 50 and the maximum number of iterations as 200 based on recommendations of previous work [24], [27]. The probabilities of GEP operators are set as follows: All of the mutation operators (i.e. Point-mutation, inversion-mutation, IS mutation and RIS mutation) rates are set to 0.1. The 1-point crossover rate is set to 0.6 and 2-point crossover is set to 0.4. All experiments are conducted on computers with Core i5-11400 2.60-GHz CPU.

TABLE II
Details of The Instance Settings

	region size	tasks	charging stations	UAVs
S1	15 × 15	15	3	1
S2	20 × 20	25	9	4
S3	15 × 15	36	6	3
S4	20 × 20	40	9	4
S5	30 × 30	50	12	5
S6	30 × 30	65	9	6
S7	30 × 30	85	16	7
S8	30 × 30	100	16	8
S9	30 × 30	138	16	8
S10	30 × 30	350	36	26
S11	30 × 30	482	20	30

B. Comparative algorithms

In this work, we test the efficiency of GP by comparing it with some bi-level optimization methods. For the upper-level task assignment problem, the individual-based meta-heuristic algorithms (TS, SA-VND [28]) and population-based meta-heuristic algorithms (GA-TS and cluster-based GA (CGA)) are considered for comparison. For the lower-level recharging scheduling problem, both enumeration and meta-heuristic methods are too time consuming. Jia et al. [12] proposed the removal-heuristic (RH) charging algorithm and compared with greedy-heuristic (GH) and forward-heuristic (FH). Since the complexity rank of these 3 heuristics is GH > RH > FH, but the effectiveness rank of them is RH > GH > FH. From the perspective of effectiveness and efficiency, this paper only considers the comparison of two heuristic algorithms, RH and FH. Overall, by combining the algorithms of the upper and lower levels, we get 8 combinations and compare them with GP.

For the fairness of the comparison, we provide the same setting for the common parameters of compared algorithms and the same number of the evaluations. The GA parameters of population size NP, crossover probability Pc and mutation probability Pm are set as 50, 0.6 and 0.1, respectively. The TS parameters of candidate set length is set as 50, taboo table length is set as 10. The other parameter settings of compared algorithm such as SA-VND all follow their original works.

C. Experiment Results

The comparison of the 9 methods over 10 independent conducted on 11 instances are shown in Table IV, where each row represents a instance. Firstly, we test the algorithms' capacity to find feasible solutions. The details are given in Table III. Then the values of feasible solutions found by each algorithm are analyzed and compared.

TABLE III
THE NUMBER OF EXPERIMENTS FOR COMPARATIVE ALGORITHMS TO FIND THE FEASIBLE SOLUTION

	CGA FH	CGA RH	TS FH	TS RH	SA-VND FH	SA-VND RH	GA-TS FH	GA-TS RH	GP
S1	10	10	10	10	10	10	10	10	10
S2	10	10	10	10	10	10	10	10	10
S3	10	10	10	10	10	10	10	10	10
S4	10	10	10	10	10	10	10	10	10
S5	8	8	10	10	10	10	10	10	10
S6	3	0	10	9	8	10	9	10	10
S7	9	0	10	10	10	10	10	10	10
S8	5	0	10	10	10	10	10	10	10
S9	4	0	10	5	9	4	10	9	10
S10	0	0	10	9	9	3	10	10	10
S11	0	0	10	0	0	0	10	5	10

1) Comparison of Ability to Find Feasible Solutions:

In Table III, we show the number of trials for all algorithms to find a feasible solution on each instance. As we can see, all algorithms can produce feasible solutions when the task number is small than 40 (S1-S4). When the problem scale increases, the search space becomes huge, some algorithms start to degenerate. Among them, the CGA divides tasks into l groups, where l is the number of UAVs. Tasks within each group are completed by one UAV. However, grouping strategy is not adoptable for TAPCS problem because sometimes tasks in groups that are too far away from the UAV or charging stations may make it difficult to find a feasible solution. In Table III, CGA-FH can only find a few feasible solutions on S5-S9 instances. Even on S10-S11 instances, it has failed. Similarly, CGA-RH is failed from the 6th instance. It can be seen that CGA algorithm based on clustering and grouping is not suitable for solving the TAPCS problem. TS-RH has successfully provided feasible solutions in most instances, but it also fails the largest one (S11). SA-VND-FH and SA-VND-RH are also in a situation where just a few or no feasible solutions can be found. In addition, for the same upper-level algorithm, the ability to find a feasible solution varies greatly when experimenting with different lower-level charging methods. FH is still effective in large instances (such as TS-FH), while RH makes the algorithm unable to find a feasible solution. It can be seen that the charging algorithm has a very important impact on the final solution. However, compared with these algorithms, only two algorithms, TS-FH and GP have clearly demonstrated their advantages in different instances. They have found feasible solutions successfully in all instances. This comparison preliminarily implies that $\Gamma(\cdot)$ evolved by the GP algorithm can stably find feasible solutions in different instances.

2) Comparison of the Objective Values:

In Table IV, the best min and mean values are highlighted. we mark "-" to denote that the algorithm can not find feasible solution. The performances are compared with GP using Wilcoxon rank-sum test if the algorithm can find feasible solutions in all 10 independent runs. The significance level is set to 0.05 with Bonferroni correction. If the algorithm cannot find all feasible solutions, we can conclude that it is worse than GP. From the results, we can get the following information.

- According to the statistic test, on small-scale instances (S1-S7), GA is only worse than CGA-RH and SA-VND-RH on S4. On the other small-scale instances, GP either performs equally well as the compared algorithms or significantly outperforms them.
- On middle-scale instances (S8-S9), GP has updated the best solutions and achieves significant results, performing better than all other algorithms. Although the average value of feasible solutions achieved by CGA-FH on S9 outperforms that of GP, it is worth noting that only 4 out of 10 experiments conducted by CGA-FH resulted in feasible solutions. Such a limited success rate reveals a significant inadequacy in CGA-FH's ability to find feasible solutions, leading to a reliability rate of merely 40%.
- On large-scale instances (S10-S11), GP demonstrates a significant performance improvement over other algorithms, resulting in an average objective value reduction of 30% or more. This shows that the rules evolved by GP can find solutions that may not be apparent or intuitive to human designers can. The flexibility in choosing between tasks and charging stations makes GP more advantageous in solving complex large-scale TAPCS problems.

3) Comparison of the heuristic charging algorithm:

To further illustrate the difference between the rules found by GP and human-designed heuristic algorithm, we show the solutions generated by FH, RH and GP on S1, as shown in the Fig. 3. The order in which tasks are executed and the placement of charging stations are closely tied to the chosen charging algorithm. In terms of FH, it's understandable that the placement of charging stations may not be optimal as it only recharges UAV when UAV can not go to the next task. From Fig. 3 (a), it can be seen that once the UAV completes task t_{12}, it will give priority to completing task t_{10}, and go to charging station c_1 to charge because the remaining battery power is not enough to go to the next task. In Fig 3 (b), using RH, the UAV opts to complete task t_{10} before heading to charging station c_1 for a recharge. This decision is based on the cost-saving analysis, which shows that removing the charging station between tasks t_3 and t_{10} yields greater cost savings compared to removing the station between t_{10} and t_{15}. Fig. 3 (c) showcases the superiority of the $\Gamma(\cdot)$ as it considers the UAV's current battery power and other relevant information. This allows for a more flexible and efficient determination of the flight route, ultimately resulting in the optimal solution.

In the feasible solution obtained by the UAV heuristic navigation model based on the GP rule, we attempted to remove the charging station in the solution. We then employed the FH and RH charging strategies to reconstruct the route. However, despite our efforts, we were unable to obtain a

TABLE IV
COMPARED FEASIBLE RESULTS OVER 10 INDEPENDENT RUNS ON 11 INSTANCES

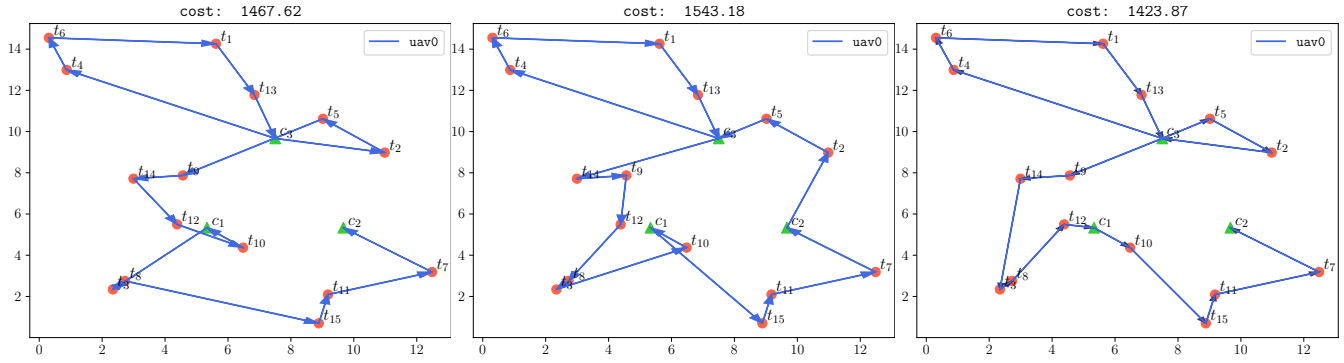
		CGA-FH	CGA-RH	TS-FH	TS-RH	SA-VND-FH	SA-VND-RH	GA-TS-FH	GA-TS-RH	GP
S1	best	1487.77	1573.06	1654.58	1651.3	1691.17	1543.18	1467.62	1477.30	**1423.87**
	mean	1607.49 ↕	1698.75 ↕	1789.52 ↑	1846.22 ↑	1818.60 ↑	1795.53 ↑	1605.82 ↕	1633.26 ↕	**1592.81**
	std	66.83	137.99	106.86	95.74	134.92	104.52	73.4	127.66	108.58
S2	best	1456.74	1327.24	1198.84	1141.92	1265.26	1185.95	1149.89	**1096.85**	1144.72
	mean	1520.58 ↑	1396.99 ↑	1427.01 ↑	1284.78 ↑	1462.2 ↑	1286.62 ↑	1207.29 ↑	**1136.79** ↕	1158.01
	std	36.03	53.94	113.74	73.04	117.7	86.62	59.09	52.85	19.04
S3	best	1113.08	1150.62	1247.16	1182.7	1246.29	1183.52	1207.58	**1090.82**	1095.56
	mean	1195.01 ↕	1211.44 ↑	1346.6 ↑	1243.97 ↑	1358.52 ↑	1230.31 ↑	1390.18 ↑	1221.55 ↕	**1146.37**
	std	41.12	26.47	71.78	71.37	73.41	28.62	122.77	103.72	52.66
S4	best	1570.02	1592.89	1842.7	1673.96	1753.49	1609.5	1608.35	**1420.94**	1740.58
	mean	1737.41 ↕	1720.30 ↓	2009.36 ↑	1786.28 ↕	1915.24 ↑	**1691.77** ↓	1865.57 ↕	1745.68 ↕	1802.01
	std	124.47	90.24	94.16	68.86	141.47	35.43	154.95	148.77	46.45
S5	best	**2646.77**	3245.38	3543.13	3310.78	2735.51	3310.35	3103.85	3226.45	2885.44
	mean	4292.5 ↑	3951.58 ↑	4063.2 ↑	3559.76 ↕	3451.34 ↕	3975.98 ↑	3691.56 ↕	3411.66 ↕	**3405.19**
	std	954.02	368.41	236.47	157.21	787.23	399.26	418.32	135.38	352.87
S6	best	3876.21		5234.83	4294.1	4545.33	4281.73	4129.41	3663.56	**3410.86**
	mean	5170.55 ↑	-	5603.64 ↑	4636.02 ↕	5281.9 ↑	4743.27 ↕	4512.78 ↑	**4338.73** ↕	4480.84
	std	808.14		222.18	220.44	429.97	321.5	284.08	438.56	646.44
S7	best	**3210.53**		5407.4	4357.71	3404.33	3670.57	4363.69	3721.8	3252.44
	mean	4423.58 ↑	-	6148.92 ↑	4632.47 ↑	4310.78 ↑	4518.23 ↑	4715.42 ↑	4025.39 ↑	**3572.57**
	std	887.44		370.35	226.67	730.36	592.31	290.38	283.96	261.42
S8	best	3446.43		6267.84	5001.56	3766.37	5083.43	5145.82	4050.82	**3443.46**
	mean	3793.28 ↑	-	6844.64 ↑	5298.53 ↑	5244.11 ↑	5560.23 ↑	5642.43 ↑	4434.84 ↑	**3732.4**
	std	228.94		314.21	172.46	1139.05	328.55	293.92	267.46	278.69
S9	best	4552.25		9385.06	7424.79	4859.01	6809.73	7157.26	5916.33	**4546.3**
	mean	**4631.88** ↑	-	10331.34 ↑	7808.87 ↑	6215.98 ↑	7311.42 ↑	7689.89 ↑	6384.27 ↑	5089.79
	std	124.38		423.24	419.3	1091.01	570.25	272.26	329.61	308.53
S10	best			16942.64	12895.49	10033.81	8922.75	12669.88	9980.31	**8024.86**
	mean	-	-	17457.17 ↑	13410.49 ↑	10730.68 ↑	9850.17 ↑	13286.34 ↑	10472.27 ↑	**8499.84**
	std			266.73	283.78	475.7	716.99	404.02	316.64	526.24
S11	best			27625.14				20029.83	16861.53	**7837.62**
	mean	-	-	28592.04 ↑	-	-	-	21423.01 ↑	17503.97 ↑	**10473.41**
	std			681.42				511.39	443.78	2416.27
w/t/l		8/3/0	9/1/1	11/0/0	8/3/0	10/1/0	9/1/1	8/3/0	5/6/0	

Columns represent different algorithms and rows represent different instances. The '-' indicates that the solution optimized by this algorithm is infeasible. ↑ denotes that GP is significantly better than the compared algorithm, ↓ denotes that GP is significantly worse than the compared algorithm, and ↕ denotes that GP is equal to the compared algorithm. "w/t/l" shows on how many instances GP wins, ties, or loses to the compared algorithm. When the algorithm fails, it loses to GP obviously.

feasible solution. Further clarification states that the human-designed heuristic charging methods have some limitations and may miss a more reasonable task execution order and recharging schedules.

V. CONCLUSION

The aim of this paper is to efficiently and effectively solve the multi-UAV power inspection problem with charging stations. First, we formulated the problem as a task assignment problem with charging stations (TAPCS) and mathematically modeled it. We then proposed a new heuristic navigation model in which all UAVs can follow a heuristic rule to complete all tasks. To improve the effectiveness of rule, we incorporate relevant information about the UAV and tasks into the terminal set and adapt GP to evolve the optimal rule $\Gamma(\cdot)$. Finally, the experimental results demonstrated the effectiveness of the proposed method on 11 instances. In the future, we plan to extend our study by investigating the TAPCS problem model and the characteristics of complex heterogeneous UAVs in real-world scenarios. This will allow us to design more complex and efficient heuristic rules.



(a) GA-TS-FH (b) SA-VND-RH (c) GP

Fig. 3. Solutions generated by FH, RH and GP on S1. The green triangles in (a), (b) and (c) represent the charging stations and the red dots represent the tasks.

REFERENCES

[1] M. Jung and H. Oh, "Heterogeneous mission planning for a single unmanned aerial vehicle (uav) with attention-based deep reinforcement learning," *PeerJ Computer Science*, vol. 8, p. e1119, 2022.

[2] B. Fei, D. Liu, J. Zhou, W. Bao, F. Chen, and H. Zhang, "A spectral clustering enabled dynamic task allocation approach of multiple uavs," in *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*. IEEE, 2022, pp. 408–415.

[3] F. Shuang, X. Chen, Y. Li, Y. Wang, N. Miao, and Z. Zhou, "Ple: Power line extraction algorithm for uav-based power inspection," *IEEE Sensors Journal*, vol. 22, no. 20, pp. 19 941–19 952, 2022.

[4] Y. Luo, X. Yu, D. Yang, and B. Zhou, "A survey of intelligent transmission line inspection based on unmanned aerial vehicle," *Artificial Intelligence Review*, vol. 56, no. 1, pp. 173–201, 2023.

[5] Z.-X. Zhang, W.-N. Chen, W. Shi, S.-W. Jeon, and J. Zhang, "An individual evolutionary game model guided by global evolutionary optimization for vehicle energy station distribution," *IEEE Transactions on Computational Social Systems*, 2023.

[6] Z. Sun, G. G. Yen, J. Wu, H. Ren, H. An, and J. Yang, "Mission planning for energy-efficient passive uav radar imaging system based on substage division collaborative search," *IEEE Transactions on Cybernetics*, 2021.

[7] F. Sun, X. Wang, and R. Zhang, "Task scheduling system for uav operations in agricultural plant protection environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2020.

[8] G. Zhenfeng, L. Yang, J. Xiaodan, and G. Sheng, "The electric vehicle routing problem with time windows using genetic algorithm," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, 2017, pp. 635–639.

[9] X.-Q. Guo, W.-N. Chen, F.-F. Wei, W.-T. Mao, X.-M. Hu, and J. Zhang, "Edge–cloud co-evolutionary algorithms for distributed data-driven optimization problems," *IEEE Transactions on Cybernetics*, 2022.

[10] F.-F. Wei, W.-N. Chen, Q. Yang, J. Deng, X.-N. Luo, H. Jin, and J. Zhang, "A classifier-assisted level-based learning swarm optimizer for expensive optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 219–233, 2020.

[11] B. Zhao, X. Liu, A. Song, W.-N. Chen, K.-K. Lai, J. Zhang, and R. H. Deng, "Primpso: A privacy-preserving multiagent particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, 2022.

[12] Y.-H. Jia, Y. Mei, and M. Zhang, "A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 855–10 868, 2021.

[13] Y. Li, M. Han, Z. Yang, and G. Li, "Coordinating flexible demand response and renewable uncertainties for scheduling of community integrated energy systems with an electric vehicle charging station: A bi-level approach," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 4, pp. 2321–2331, 2021.

[14] Y.-H. Jia, Y. Mei, and M. Zhang, "Confidence-based ant colony optimization for capacitated electric vehicle routing problem with comparison of different encoding schemes," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1394–1408, 2022.

[15] F.-F. Wei, W.-N. Chen, Q. Li, S.-W. Jeon, and J. Zhang, "Distributed and expensive evolutionary constrained optimization with on-demand evaluation," *IEEE Transactions on Evolutionary Computation*, 2022.

[16] F.-F. Wei, W.-N. Chen, and J. Zhang, "A hybrid regressor and classifier-assisted evolutionary algorithm for expensive optimization with incomplete constraint information," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[17] X.-C. Liao, W.-N. Chen, X.-Q. Guo, J. Zhong, and X.-M. Hu, "Crowd management through optimal layout of fences: An ant colony approach based on crowd simulation," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[18] S. Karakatič, "Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm," *Expert Systems with Applications*, vol. 164, p. 114039, 2021.

[19] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023.

[20] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022.

[21] X.-C. Liao, W.-J. Qiu, F.-F. Wei, and W.-N. Chen, "Combining traffic assignment and traffic signal control for online traffic flow optimization," in *International Conference on Neural Information Processing*. Springer, 2022, pp. 150–163.

[22] A. Moadab, F. Farajzadeh, and O. Fatahi Valilai, "Drone routing problem model for last-mile delivery using the public transportation capacity as moving charging stations," *Scientific Reports*, vol. 12, no. 1, pp. 1–16, 2022.

[23] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[24] C. Ferreira, "Gene expression programming: a new adaptive algorithm for solving problems," *arXiv preprint cs/0102027*, 2001.

[25] Y.-H. Jia, Y. Mei, and M. Zhang, "A memetic level-based learning swarm optimizer for large-scale water distribution network optimization," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 1107–1115.

[26] X. Yu, W.-N. Chen, X.-M. Hu, T. Gu, H. Yuan, Y. Zhou, and J. Zhang, "Path planning in multiple-auv systems for difficult target traveling missions: a hybrid metaheuristic approach," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 12, no. 3, pp. 561–574, 2019.

[27] X.-C. Liao, Y.-H. Jia, X.-M. Hu, and W.-N. Chen, "Uncertain commuters assignment through genetic programming hyper-heuristic," *IEEE Transactions on Computational Social Systems*, 2023.

[28] H. Liu, X. Li, G. Wu, M. Fan, R. Wang, L. Gao, and W. Pedrycz, "An iterative two-phase optimization method based on divide and conquer framework for integrated scheduling of multiple uavs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5926–5938, 2020.