electronics

MDPI

Article

Ability-Restricted Indoor Reconnaissance Task Planning for Multiple UAVs

Ruowei Zhang [1], Lihua Dou [1], Qing Wang [1,*], Bin Xin [1,*] and Yulong Ding [2]

[1] School of Automation, Beijing Institute of Technology, Beijing 100081, China
[2] Peng Cheng Laboratory, Shenzhen 518055, China
* Correspondence: wangqing1020@bit.edu.cn (Q.W.); brucebin@bit.edu.cn (B.X.)

Abstract: For indoor multi-task planning problems of small unmanned aerial vehicles (UAVs) with different abilities, task assignment and path planning play a crucial role. The multi-dimensional requirements of reconnaissance tasks bring great difficulties to the task execution of multi-UAV cooperation. Meanwhile, the complex internal environment of buildings has a great impact on the path planning of UAVs. In this paper, the ability-restricted indoor reconnaissance task-planning (ARIRTP) problem is solved by a bi-level problem-solving framework. In the upper level, an iterative search algorithm is used to solve the task assignment problem. According to the characteristics of the problem, a solution-space compression mechanism (SSCM) is proposed to exclude solutions that do not satisfy the task requirements. In the lower level, based on a topological map, the nearest neighbor (NN) algorithm is used to quickly construct the path sequence of a UAV. Finally, the genetic algorithm (GA) and simulated annealing (SA) algorithm are applied to the upper level of the framework as iterative search algorithms, which produces two hybrid algorithms named the GA-NN and SA-NN, respectively. ARIRTP instances of different scales are designed to verify the effectiveness of the SSCM and the performance of the GA-NN and SA-NN methods. It is demonstrated that the SSCM can significantly compress the solution space and effectively improve the performance of the algorithms. The proposed bi-level problem-solving framework provides a methodology for the cooperation of multi-UAV to perform reconnaissance tasks in indoor environments. The experimental results show that the GA-NN and SA-NN methods can quickly and efficiently solve the ARIRTP problem. The performance of the GA-NN method is similar to that of the SA-NN method. The GA-NN method runs slightly faster. In large-scale instances, the performance of the SA-NN method is slightly better than that of the GA-NN method.

Keywords: reconnaissance task planning; unmanned aerial vehicles; topological map; indoor environment; genetic algorithm; simulated annealing algorithm; nearest neighbor algorithm

check for updates

Citation: Zhang, R.; Dou, L.; Wang, Q.; Xin, B.; Ding, Y. Ability-Restricted Indoor Reconnaissance Task Planning for Multiple UAVs. *Electronics* **2022**, *11*, 4227. https://doi.org/10.3390/electronics11244227

Academic Editor: Nurul I. Sarkar

Received: 14 November 2022
Accepted: 11 December 2022
Published: 19 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Due to the advantages of flexibility and mobility, unmanned aerial vehicles (UAVs) are increasingly used in various environments [1–11]. In mountainous environments, UAVs are widely used in searches, reconnaissance, surveillance, fault detection, data collection, target recognition, and classification [12–16]. In suburban environments, UAVs are widely used in agriculture such as for watering, sowing, and spraying pesticides [17]. In urban environments, UAVs are widely used in searches, reconnaissance, and other tasks. Recently, there has been an increasing demand for UAVs to solve complex indoor tasks such as target searches [18], surveillance, search and rescue [19], package delivery to large indoor facilities, and task-scheduling systems [20].

Compared with outdoor environments, such as urban environments and mountainous environments, indoor environments have more restrictions on UAVs [21]. In outdoor environments, there may be no-fly areas and dangerous areas because of bad weather or electromagnetic interference [22], which require UAVs to avoid those areas or pass

through those areas in the shortest possible time. Obstacle avoidance is an important problem for UAVs in path planning. In contrast, such constraints in indoor environments may be different. Specifically, for mobility, UAVs cannot move forward smoothly in an indoor environment because of the narrow indoor space and safe flight distance of UAVs. In addition, compared with outdoor open environments, the connection relationships between indoor rooms are complex and have a serious impact on the path planning of UAVs.

In existing research on UAVs, the task assignment and path planning of UAVs are often studied separately [23,24]. In [23], an improved algorithm based on the simulated annealing algorithm (SA), namely the swap-and-judge simulated annealing (SJSA) algorithm is used to solve the multi-task planning problem of UAVs. However, in this problem, the task requirements and UAV abilities are single. The impact of environmental factors on UAV path planning is not specifically considered. In [25], a bi-level ant colony optimization (BACO) algorithm was proposed to solve the capacitated electric vehicle routing problem (CEVRP). The upper level determines the visiting sequence of the tasks, and the lower level adjusts the task sequence generated by the upper level according to the electricity constraint. In addition, most of the existing research on indoor path planning is to find a path that meets the requirements so that the UAV can smoothly reach the destination from the starting point [21,26]. Most studies aim to find an optimal path for one UAV. There are also some studies focusing on multi-UAV cooperative path planning but the process of task assignment is not considered [27,28]. In recent studies, most reconnaissance tasks have single requirements and the abilities of UAVs are also single. When the multi-task assignments of multiple UAVs are carried out, these problems can be modeled as the classical vehicle routing problem (VRP). However, when reconnaissance tasks have multi-dimensional requirements and the abilities of UAVs are heterogeneous, it may be necessary to use multiple UAVs with different abilities to complete a single reconnaissance task.

Motivated by the above observations, in order to balance the multi-task assignment and path planning of multiple UAVs, this paper considers the multi-dimensional requirements of reconnaissance tasks. Meanwhile, the influence of the connection relationships between rooms on the path planning of UAVs is considered based on the indoor topological map. This paper focuses on solving the ability-restricted indoor reconnaissance task-planning (ARIRTP) problem. The main contributions of this paper are as follows:

- The ARIRTP problem is modeled as a combinatorial optimization problem in which the requirements of the reconnaissance tasks and the abilities of UAVs have multiple dimensions. A topological map is used to present the connection relationships between indoor rooms and help UAVs to conduct global path planning;
- A bi-level problem-solving framework is proposed for the ARIRTP problem. The upper level uses an iterative search algorithm to solve the task assignment problem of UAVs. According to the characteristics of the problem, a solution space compression mechanism is proposed to make the generated task assignment schemes meet the requirements of the reconnaissance tasks. The lower level uses the nearest neighbor (NN) algorithm to quickly construct the path sequence of each UAV based on the topology information of buildings;
- Two hybrid algorithms are proposed by applying the genetic algorithm (GA) and SA algorithm as the iterative search algorithms to the upper level of the bi-level problem-solving framework, respectively.

The remainder of this paper is organized as follows. The problem description and the mathematical model are described in Section 2. The details of the proposed algorithms are described in Section 3. The discussions of the design of the experiments and the experimental results are given in Section 4. Finally, Section 5 provides the conclusion and perspectives.

2. Problem Formulation

In this section, first, the problem description of ARIRTP is given. Then, the environment modeling process is described. Finally, the problem model is presented.

2.1. Problem Description

A group of UAVs equipped with various sensors is required to conduct a set of reconnaissance tasks with different requirements distributed in a building. In an indoor environment, multiple rooms are connected by doors. A series of reconnaissance tasks with multiple dimensions and different requirements are scattered indoors. Based on the requirements of the reconnaissance tasks, we judge whether the UAVs are able to complete the whole task or part of the task. If the UAVs cannot independently meet the requirements of all dimensions of a certain reconnaissance task, multiple UAVs need to be combined to make their common capabilities meet the reconnaissance requirements of the whole task.

Figure 1 shows the situation of multiple reconnaissance task planning for multiple UAVs in an indoor environment. Three UAVs are required to take off from the entrance point (start point) and enter the building to perform seven reconnaissance tasks. If one UAV can meet the requirements of all dimensions of a task, it can complete the task independently. For example, UAV U_1 independently performs reconnaissance tasks T_7 and T_6. However, when the UAV cannot meet all the requirements of the task alone, it needs to complete the corresponding reconnaissance task cooperatively in a certain combination mode. For instance, UAV U_2 and UAV U_3 cooperate to conduct reconnaissance task T_1 because they cannot complete the task alone. When UAV U_2 and UAV U_3 pass through door D_4, the two UAVs may collide. Now, UAV U_2 can choose to wait for UAV U_3 to pass through door D_4 first according to the priority of the task to avoid a collision. Topological maps represent the environment using graphs, where the vertices and edges represent the rooms and connection relationships between the rooms, respectively [29–32].

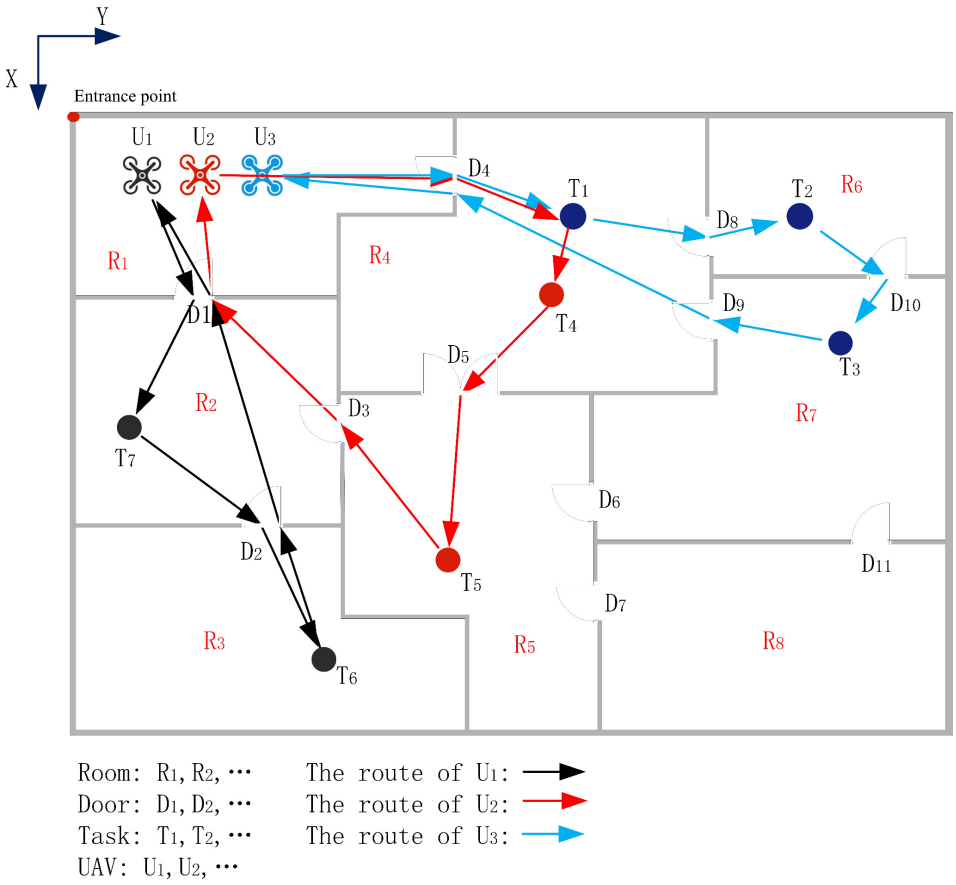

Figure 1. An example of three UAVs performing seven reconnaissance tasks in an indoor environment.

Due to ability restrictions, a variety of task assignment situations could occur. One UAV performs one reconnaissance task, one UAV performs multiple reconnaissance tasks, or multiple UAVs perform one reconnaissance task.

2.2. Environment Modeling

The contents of the indoor environment are modeled as follows: the index sets of rooms, doors, task points, and UAVs are denoted by R_n, D_n, T_n, and U_n, respectively. The coordinate matrices of the doors and reconnaissance task points are described by P_D and P_T, where x and y represent the horizontal and vertical coordinates of the different doors or task points.

$$R_n = \{R_1, R_2, \cdots, R_{N_R}\}. \tag{1}$$

$$D_n = \{D_1, D_2, \cdots, D_{N_D}\}. \tag{2}$$

$$T_n = \{T_1, T_2, \cdots, T_{N_T}\}. \tag{3}$$

$$U_n = \{U_1, U_2, \cdots, U_{N_U}\}. \tag{4}$$

$$P_D = \begin{bmatrix} x_{D_1} & y_{D_1} \\ x_{D_2} & y_{D_2} \\ \cdots & \cdots \\ x_{D_{N_D}} & y_{D_{N_D}} \end{bmatrix}. \tag{5}$$

$$P_T = \begin{bmatrix} x_{T_1} & y_{T_1} \\ x_{T_2} & y_{T_2} \\ \cdots & \cdots \\ x_{T_{N_T}} & y_{T_{N_T}} \end{bmatrix}. \tag{6}$$

The vector R indicates the room the task points are in. r_j represents the index number of the room that task j belongs to.

$$R = (r_1, r_2, \cdots, r_j, \cdots, r_{N_T}), r_j \in R. \tag{7}$$

The connectivity matrix between rooms is described by A. If two rooms share a door, the two rooms are connected.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N_R} \\ \cdots & \cdots & a_{i,j} & \cdots \\ a_{N_R,1} & a_{N_R,2} & \cdots & a_{N_R,N_R} \end{bmatrix}. \tag{8}$$

$$a_{i,j} = \begin{cases} 0 & \text{if } i = j, \quad \forall i, j \in R, \\ g & \text{if room } i \text{ and room } j \text{ are directly connected,} (g \in D), \\ \infty & \text{if room } i \text{ and room } j \text{ are indirectly connected.} \end{cases} \tag{9}$$

The distance matrix between two doors is described by B. If the two doors are in the same room, the distance between the doors can be approximately represented by the Euclidean distance of the positions of the doors (represented by $distance(D_h, D_g)$). If the two doors are not in the same room, the distance between the doors can be obtained based on the Dijkstra algorithm (represented by $distance_{Dij}(D_h, D_g)$). $argmin(distance_{Dij}(D_h, D_g))$ represents the coordinate matrix of the doors connecting door h and door g.

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,N_D} \\ \cdots & \cdots & b_{h,g} & \cdots \\ b_{N_D,1} & b_{N_D,2} & \cdots & b_{N_D,N_D} \end{bmatrix}. \tag{10}$$

$$b_{h,g} = \begin{cases} 0 & \text{if } h = g, \quad \forall \quad h, g \in D, \\ distance(D_h, D_g) & \text{if door } h \text{ and door } g \text{ are in one same room,} \\ distance_{Dij}(D_h, D_g) & \text{if door } h \text{ and door } g \text{ are not in one same room.} \end{cases} \tag{11}$$

$$argmin(distance_{Dij}(D_h, D_g)) = \begin{bmatrix} x_{D_h} & y_{D_h} \\ \cdots & \cdots \\ x_{D_g} & y_{D_g} \end{bmatrix}. \quad (12)$$

2.3. Problem Model

In this section, the ARIRTP problem model is given in detail. First, the related assumptions are made. Then, the constraints and decision variables in the ARIRTP problem are represented. Finally, the objective function is described. The list of relevant parameters and variables is shown in Table 1.

Table 1. List of parameters and variables.

Symbols	Description
N_U	Total number of UAVs
N_T	Total number of tasks
N_S	Total number of sensor types
i	Index of UAVs
j, q	Index of tasks
h, g	Index of doors
n	Index of sensor types
k	Index of the combination mode of UAVs
l_i	Tour length of UAV i
e_i	Maximum endurance of UAV i
θ_j^n	Threshold of the demand on the n-th sensor type in task j
u_i^n	Ability value of the n-th sensor type for UAV i
L_i	Task list of UAV i
P_i	Task sequence of UAV i
m_i	Total number of tasks assigned to UAV i
$d_{j,q}$	Distance between the position of task j and task q
$d_{j,h}$	Distance between the position of task j and door h
$d_{h,g}$	Distance between the position of door h and door g
$d_{g,q}$	Distance between the position of door g and task q
$X = [x_{i,j}]_{N_U \times N_T}$	Assignment scheme
$x_{i,j} \in \{0, 1\}$	Task j is assigned to UAV i or not
$L = \{ L_1; L_2; \cdots ; L_{N_U} \}$	Task list of total UAVs
$P = \{ P_1; P_2; \cdots ; P_{N_U} \}$	Task sequence set of total UAVs
$\mathcal{I}_\mathcal{U}$=$\{1, 2, \cdots, N_U\}$	Index set of the UAV
$\mathcal{I}_\mathcal{T}$=$\{1, 2, \cdots, N_T\}$	Index set of the task
$\mathcal{I}_\mathcal{S}$=$\{1, 2, \cdots, N_S\}$	Index set of the sensor type

2.3.1. Assumptions

The assumptions about the relevant situations in the ARIRTP problem are made from three aspects: the UAVs, reconnaissance tasks, and indoor environment [33–36].

- UAVs
  (1) The curvature constraint of the UAVs is not considered because small-rotor UAVs are usually used in indoor environments and their flight speed is slow.
  (2) The collision avoidance problem of UAVs is ignored. If the trajectories of the UAVs have intersections, the UAV with a low task priority waits for the UAV with a high task priority to pass through the place where a collision may occur. Then, the UAV with the low task priority starts to perform its task again.
  (3) All the UAVs fly at the same constant speed.
  (4) The communication between the UAVs is ignored.

- Reconnaissance tasks
  (1) The reconnaissance tasks do not require the UAVs to perform the tasks synchronously.
  (2) The UAVs are not required to reach and leave the task points at the same time.
  (3) The reconnaissance time can be ignored.
  (4) If two task points are distributed in the same room, the distance between these two task points can be approximately expressed by the Euclidean distance. If two task points are distributed in different rooms that can be connected directly, the distance between the two task points can be approximately expressed by the sum of $d_{j,h}$ and $d_{h,q}$. If two task points are distributed in different rooms that can be connected indirectly, the distance between the two task points can be approximately expressed by the sum of $d_{j,h}$, $d_{h,g}$, and $d_{g,q}$. If two task points are distributed in two completely disconnected rooms, the distance between the two task points can be defined as infinity.
- Indoor environments
  (1) The information on the indoor map is known a priori.
  (2) Two connected rooms share only one door.
  (3) The rooms inside the building are fully connected, that is, a room can reach any other room through a certain topological relationship.
  (4) The no-fly zone and interference zone for the UAVs are not considered in indoor environments.
  (5) The obstacles in indoor environments will not affect the path planning of the UAVs because they can adjust their height to avoid obstacles and the time cost caused by adjusting the height can be ignored.

2.3.2. Decision Variables

Decision variables include two parts: X and P. $X = [x_{i,j}]_{N_U \times N_T}$ represents a task assignment scheme. $x_{i,j} \in \{0, 1\}$ means reconnaissance task j is assigned to UAV i or not. $P = \{ P_1; P_2; \cdots ; P_i; \cdots ; P_{N_U} \}$ denotes the task sequence set of each UAV. P_i is shown in Formula (14). p_i^0 and $p_i^{m_i+1}$ represent the starting point and ending point, that is, the point with coordinates (0,0).

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N_T} \\ \cdots & \cdots & x_{i,j} & \cdots \\ x_{N_U,1} & x_{N_U,2} & \cdots & x_{N_U,N_T} \end{bmatrix}. \tag{13}$$

$$P_i = (p_i^0, p_i^1, \cdots, p_i^{m_i}, p_i^{m_i+1}), i \in \mathcal{I}_{\mathcal{U}}. \tag{14}$$

2.3.3. Constraints

In order to ensure the feasibility of the planned scheme, the following constraints need to be met.

• Endurance constraint

$$l_i \leq e_i, i \in \mathcal{I}_{\mathcal{U}}. \tag{15}$$

Formula (15) stipulates that the tour length of each UAV must be less than its maximum endurance.

• Ability demand constraint

$$\exists \quad i \in \mathcal{I}_{\mathcal{U}} : u_i^n x_{ij} \geq \theta_j^n, \quad \forall \quad j \in \mathcal{I}_{\mathcal{T}}, n \in \mathcal{I}_{\mathcal{S}}. \tag{16}$$

Formula (16) ensures that for each task, at least one UAV assigned to this task can meet the d-th requirement.

2.3.4. Cost Function

In this paper, the task assignment scheme and the path planning results are evaluated by constructing a specific cost function. The mathematical definition of the ARIRTP problem is shown below:

$$F(X,P) = \sum_{i=1}^{N_U} l_i. \tag{17}$$

$$l_i = d_{p_i^0,p_i^1} + d_{p_i^1,p_i^2} + \cdots + d_{p_i^{m_i-1},p_i^{m_i}} + d_{p_i^{m_i},p_i^{m_i+1}}, \quad m_i \le N_T, i \in \mathcal{I}_{\mathcal{U}}. \tag{18}$$

Formula (17) represents the total tour length of all UAVs and the objective of the ARIRTP problem is to minimize the total tour length of the UAVs while performing the tasks. Formula (18) represents the tour length of each UAV.

The ARIRTP problem model is shown below:

$$min \quad F(X,P) \quad s.t. \quad (11),(12), x_{i,j} \in \{0,1\} \tag{19}$$

3. Algorithm Design

In this section, in order to better solve the ARIRTP problem modeled in the previous section, we propose a bi-level problem-solving framework. The upper level uses an iterative search algorithm to solve the ability-restricted task assignment problem of UAVs. The results of the task assignment of the upper level are input to the lower level. Then, the lower level uses the nearest neighbor algorithm to quickly construct the task sequence of each UAV based on the topology information of buildings. Meanwhile, the upper and lower levels jointly participate in the decoding process to obtain a complete solution to evaluate the task assignment scheme generated at the upper level.

3.1. Bi-Level Problem-Solving Framework

The proposed framework is shown in Figure 2. In the upper level, an iterative search method is used to solve the task assignment problem. Common iterative search algorithms include global search algorithms and local search algorithms or exact algorithms and meta-heuristic algorithms. According to the characteristics of the ARIRTP problem, the encoding and decoding mechanisms are designed and the knowledge contained in the problem is mined. Meanwhile, a solution space compression mechanism suitable for the characteristics of the problem is proposed. After the solution space is compressed, the iterative search algorithms will use the compressed solution space to generate new solutions. All the newly generated solutions can meet the requirements of the tasks. Then, the task list assigned by the upper level for each UAV will be input to the lower level. In the lower level, the nearest neighbor algorithm is used to quickly construct the task-visiting sequences for the UAVs. In addition, the lower level also participates in the decoding process of the upper level and constructs an executable path that conforms to the topological constraints. A complete task-planning scheme is obtained through decoding by both the upper and lower levels. The termination condition will be determined according to the specific iteration search algorithm used in the upper level.

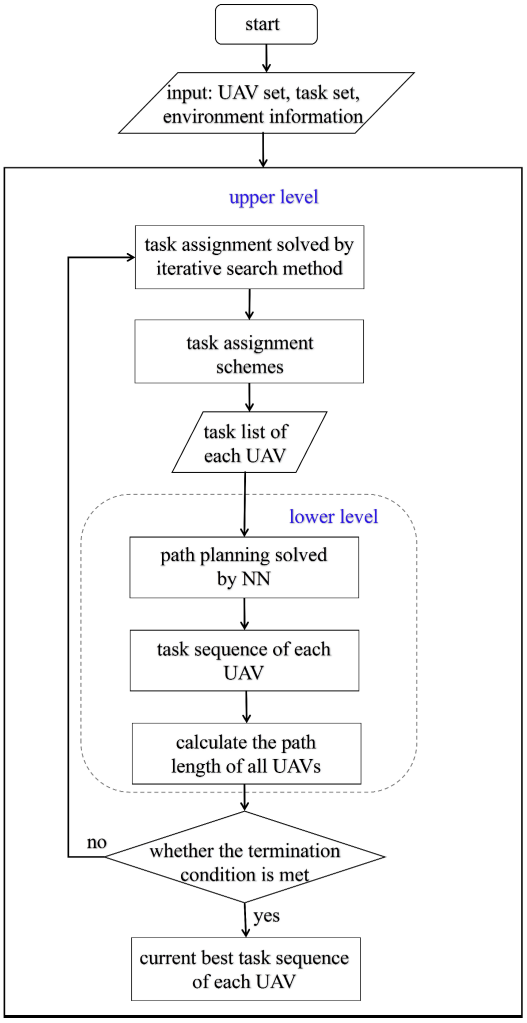

Figure 2. The bi-level problem-solving framework.

3.1.1. Task Assignment

• Encoding and decoding

Based on the encoding scheme, an individual can be described using Formula (20) and the initial population is formed by randomly generating multiple individuals.

$$S = (s_1, s_2, \cdots, s_j, \cdots, s_{N_T}), s_j \in \{1, 2, \cdots, 2^{N_u} - 1\}. \tag{20}$$

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N_U} \\ \cdots & \cdots & c_{k,i} & \cdots \\ c_{2^{N_u}-1,1} & c_{2^{N_u}-1,2} & \cdots & c_{2^{N_u}-1,N_U} \end{bmatrix} \tag{21}$$

s_j represents the index number of the combination mode of the UAVs that perform task j. The constant matrix C represents all the combination modes of the UAVs. The vector $C(k,:)$ is the k-th row of C and represents the k-th combination mode and $c_{k,i} \in \{0,1\}$. The data in $C(k,:)$ are removed and expressed in binary form, recorded as b_k. b_1=1 and $b_{k+1} = b_k + 1$, $0 < k < 2^{N_u} - 1$, $k \in \mathbf{N}$.

The decoding process is shown in Table 2. For ease of understanding, an example of three UAVs and ten task points is shown in Figure 3. When the number of UAVs is determined to be 3, the corresponding constant matrix C can be determined. Referring to matrix C, the solution vector S can be converted into a matrix X and then the task list L_i of each UAV can be extracted.

Table 2. The decoding process of the upper level.

Decoding of the upper level
1. Vector S
2. Construct combination mode matrix C \\ According to Formula (17)
3. Transfer S to X referring to C \\ Obtain a complete task assignment scheme
4. Obtain L_i based on X \\ Obtain the task lists
5. Set $L_i = \emptyset$
6. For $i = 1 : N_U$
7. For $j = 1 : N_T$
8. $x_{i,j} = C(s_j, i)$
9. If $X(i,j) == 1$
10. $L_i = L_i \bigcup \{j\}$
11. End if
12. End for
13. End for
14. Output L

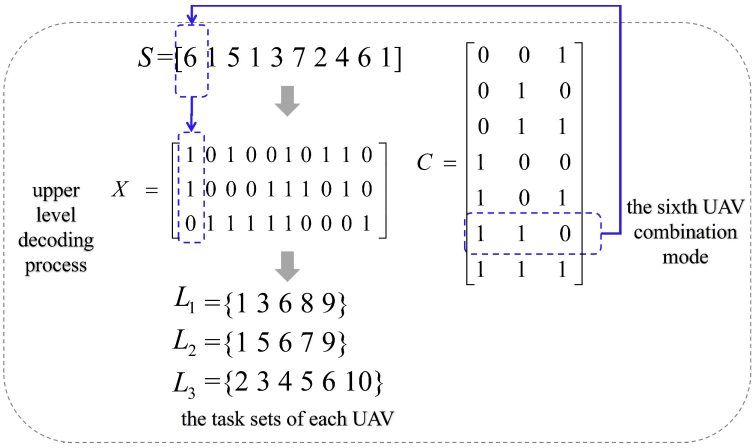

Figure 3. The decoding process of the upper level of the example of three UAVs and ten tasks.

• Solution space compression mechanism

The solution space compression mechanism is a method proposed to eliminate UAV combinations that cannot meet the task requirements. When there are more solutions that do not meet the task requirements in the instances, the effect of solution space compression will be more obvious. The algorithm flow of solution space compression is shown in Algorithm 1. First, the set $t_j = \{1, 2, \cdots, 2^{N_u} - 1\}$ is constructed, representing all the combination modes of the UAVs. Then, we judge whether the combination mode can satisfy the requirements of each task. According to Formula (16), in one combination mode, at least one UAV assigned to this task can meet the requirements; this combination mode will be retained; otherwise, it will be abandoned.

Algorithm 1 Solution space compression mechanism
1. Input set $t_j, \forall j \in \mathcal{I}_\mathcal{T}$
2. For $j = 1 : N_T$ \\ Traverse each task
3. For $k = 1 : 2^{N_U} - 1$ \\ Traverse each combination mode of UAVs
4. If $\nexists\, u_i^n \geq \theta_j^n, \forall n \in \mathcal{I}_\mathcal{S}$ \\ The UAVs in the combination mode k all cannot satisfy the
5. \\ requirements of task j
6. $t_j = t_j \setminus \{ k \}$
7. End if
8. End for
9. End for
10. Output $t_j, \forall j \in \mathcal{I}_\mathcal{T}$ \\ The compressed task list

The specific process and results of solution space compression in the example of three UAVs and ten tasks are shown in Figure 4. Since there are two options for whether the UAVs are assigned to perform tasks, the number of the whole assignment scheme is $(2^3 - 1)^{10}$. The size of the compressed solution space becomes 0.00007 times the original.

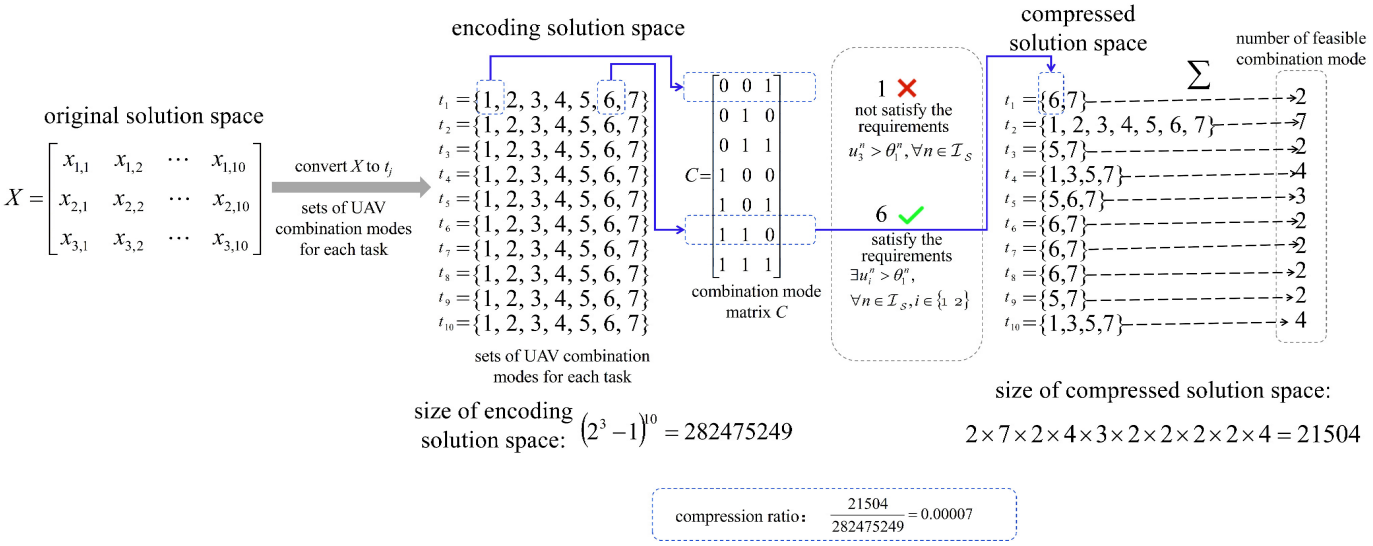

Figure 4. Example of solution space compression process of three UAVs and ten tasks.

3.1.2. Path Planning

In the lower level, based on the task list of each UAV obtained in the upper level, the path planning is carried out using the nearest neighbor algorithm to obtain the path sequence of each UAV based on an indoor topology map.

• Nearest neighbor algorithm based on topological map

In order to meet the constraints of indoor environments, we use a topology map for path planning. This is necessary to determine not only the sequences of the task points but also the sequence of the path points that can be executed by each UAV. The algorithm flow of the nearest neighbor algorithm based on a topological map is shown in Algorithm 2.

• Path sequence generation method

According to the connectivity of the indoor rooms, it needs to be judged whether any two adjacent task points in the UAV task sequence meet the constraints of the environment. According to the specific indoor environment, the following three situations are summarized: (1) if two connected task points are in the same room, they do not need to be processed; (2) if they are in two directly connected rooms, we need to add the door connecting the two rooms to the sequence, and (3) if they are in two indirectly connected rooms, we will find a connecting sequence of doors based on Formula (12) and add it to the sequence.

Remark 1. *The task list indicates the set of tasks assigned to each UAV after the task assignment. The task sequence represents the visiting sequence vector of the task points of the UAVs. The path sequence indicates the visiting waypoint vector of the UAVs.*

Algorithm 2 Nearest neighbor algorithm based on topological map

1. Input: task list L, start point p_{start}, end point p_{end}
2. $P_i = \mathbf{0}$, $P_i^{'} = \mathbf{0}$, $d = \mathbf{0}$, the final path sequence list $P_i^{'}$, total tour length $l_{total} = 0$
3. For $i = 1 : N_U$
4. $P_i = [P_i; p_{start}]$, $P_i^{'} = [P_i^{'}; p_{start}]$
5. While $L_i \neq \emptyset$
6. $p_{size} = size(P_i)$, $m_i = size(L_i)$ \\ Function $size()$ is used to find the length of L_i or P_i
7. For $j = 1 : m_i$
8. $d(j) = distance(P_i(p_{size}), L_i(j))$
9. End for
10. $p_{curr} = L_i(argmin(d))$ \\ p_{curr} is temporary point
11. $P_i = [P_i; p_{curr}]$, $P_i^{'} = [P_i^{'}; p_{curr}]$, $L_i = L_i \setminus \{p_{curr}\}$
12. $m_i = m_i - 1$, $p_{size} = p_{size} + 1$
13. If $A(R(P_i(p_{size})), R(P_i(p_{size}-1)))! = 0 \& A(R(P_i(p_{size})), R(P_i(p_{size}-1)))! = \infty$
14. \\ Two points in two directly connected rooms
15. $P_i^{'} = [P_i^{'}; P_D((A(R(P_i(p_{size})), R(P_i(p_{size}-1))))), :)]$
16. Elseif $A(R(P_i(p_{size})), R(P_i(p_{size}-1))) = \infty$ \\ Two points in two indirectly
17. \\ connected rooms
18. $P_{Dij} = argmin(distance_{Dij}(D_h, D_g))$ \\ Based on Formulas (11) and (12)
19. \\ P_{Dij} is a coordinate matrix of the doors (connecting the two points)
20. $P_i^{'} = [P_i^{'}; P_{Dij}]$
21. End if
22. End while
23. $P_i = [P_i; p_{end}]$, $P_i^{'} = [P_i^{'}; p_{end}]$
24. $p_{size} = size(P_i^{'})$
25. If $A(R(P_i(p_{size})), R(P_i(p_{size}-1)))! = 0 \& A(R(P_i(p_{size})), R(P_i(p_{size}-1)))! = \infty$
26. $P_i^{'} = [P_i^{'}; P_D((A(R(P_i(p_{size})), R(P_i(p_{size}-1))))), :)]$
27. Elseif $A(R(P_i(p_{size})), R(P_i(p_{size}-1))) = \infty$
28. $P_{Dij} = argmin(distance_{Dij}(D_h, D_g))$
29. $P_i^{'} = [P_i^{'}; P_{Dij}]$
30. End if
31. $l_i = length(P_i^{'})$, $l_{total} = l_{total} + l_i$ \\ Sequence length function $length()$ is formula (14)
32. $P_i^* = P_i^{'}$ \\ P_i^* is the visiting waypoint matrix of the i-th UAV
33. $d = \mathbf{0}$, $P_i = \mathbf{0}$, $P_i^{'} = \mathbf{0}$, $l_{total} = 0$
34. End for
35. $P^* = \{P_1^*; P_2^*; ...; P_{N_U}^*\}$ \\ P^* is the visiting waypoint matrix of all UAVs
36. Output P^*, l_{total}

3.2. Iterative Search Algorithm

The iterative search method is mainly composed of three parts: initializing solutions, evaluating solutions, and generating new solutions according to certain rules. The iterative search algorithm includes the global search algorithm and local search algorithm. As a typical global search algorithm, the GA has been widely used since it was proposed. After years of development, many studies still concentrate on improving the GA at different stages, such as the initialization stage, evolution stage, and individual evaluation, among others [37]. As another promising option, the SA [38] was one of the winning algorithms for the electric vehicle routing problem in the IEEE WCCI2020 competition [39], which shows its efficiency in solving the VRP and its variants or other combinatorial optimization problems [40,41].

Therefore, we apply the GA and SA as iterative search algorithms to the proposed bi-level problem-solving framework, which produces two hybrid algorithms named the GA-NN and SA-NN, respectively.

3.2.1. Genetic Algorithm

• Initialization stage

First, we load the reconnaissance tasks, UAVs, and indoor environment parameters. Then, the population is initialized. For the population initialization process, we use the compressed solution space to randomly generate individuals, which ensures that the UAVs can meet the requirements of each reconnaissance task in the task assignment scheme.

•Evolutionary stage

(1) Fitness evaluation

In order to ensure the algorithm searches in the expected direction, a solution needs to be properly evaluated. In this paper, the fitness function is realized by the following formula:

$$h_p^g = 1 - (f_p^g - f_{min}^g)/(f_{max}^g - f_{min}^g + \varepsilon). \tag{22}$$

h_p^g represents the fitness value of the hth individual in the gth generation and $h_p^g \in (0,1)$. f_p^g represents the objective function value of the hth individual in the gth generation. f_{min}^g and f_{max}^g represent the minimum and maximum objective function values in the gth generation, respectively. We set the parameter $\varepsilon = 0.001$ in order to avoid $f_{min}^g - f_{max}^g + \varepsilon$ =0.

(2) Operator design

Selection operator

In this paper, the objective is to minimize the total tour length of UAVs. When the objective value is smaller, the fitness evaluation value is closer to 1. So, we prefer to retain individuals with a high fitness value. For each individual, we generate a random number. If the fitness evaluation value of the individual is bigger than this random number, the individual is retained. Otherwise, the individual is abandoned, that is, if the following formula is satisfied, we retain the hth individual in the gth generation.

$$h_p^g > rand, rand \in (0,1). \tag{23}$$

Crossover operator

After the selection operation, the selected individuals form a new population, that is, the offspring population. However, the genes of each individual in the offspring population do not change. Now, in order to generate new individuals, it is necessary to change the genes of the individual. In this paper, the uniform crossover operator is used. First, two individuals are randomly selected from the population. Then, each gene of the first individual is traversed and the random number $rand_j \in (0,1)$ is generated and compared with the crossover probability p_c to determine whether or not to carry out the crossover operation. If $rand_j > p_c$, the jth gene of the first individual exchanges with the jth gene of the second individual.

Mutation operator

In this paper, the uniform mutation operator is used. First, each gene of the individual is traversed and the random number $rand_j \in (0,1)$ is generated and compared with the mutation probability p_m. Then, it is determined whether or not to carry out a mutation operation in each gene location. If $rand_j > p_m$, the jth gene of the individual mutates and the mutation value are randomly selected from t_j.

Recombination operator

We set the population size as NP. After the selection operation, some individuals are abandoned. In order to ensure that the population size is NP, the crossover and mutation operations are required to be made to generate new individuals until the population size reaches NP.

To solve the ARIRTP problem, we propose a hybrid algorithm based on the GA and the nearest neighbor algorithm. The algorithm flow is shown in Algorithm 3. The GA-NN method mainly includes three stages: the solution space compression stage, initialization stage, and evolutionary stage. In the initialization stage, the compressed solution space is used to generate new individuals. When evaluating individuals, the GA and the NN algorithm jointly participate in the decoding process. The GA is responsible for decoding

the task assignment results of the upper level. The NN algorithm is responsible for decoding the path planning results of the lower level.

Algorithm 3 GA-NN

1. Input: instances
2. Set iteration number $gen = 1$, maximum number of iterations $Maxgen$, population size NP
3. Selected population $P_{sel} = \emptyset$, population size $P_{size} = 0$
4. Solution space compression \\ According to Section 3.1.1
5. Initialize current population P \\ Task assignment
6. While $gen \leq Maxgen$ \\ Main loop
7. For $i = 1 : NP$
8. $r = rand$ \\ Rand number $rand \in (0, 1)$
9. $f(i) = F(P(i))$ \\ Objective value of individual i, according to Equations (17) and (18) and Algorithm 2
10. $h(i) = fitness(f(i))$ \\ Fitness value of individual i, according to Equation (22)
11. If $h(i) > r$
12. $P_{sel} = P_{sel} \cup \{P(i)\}$
13. End if
14. End for
15. $P = \emptyset$, $P = P_{sel}$, $P_{sel} = \emptyset$
16. $P_{size} = size(P)$
17. While $P_{size} < NP$
18. $p1 = rand * P_{size}$, $p2 = rand * P_{size}$ \\ Two individuals are randomly selected
19. $(p1, p2) = crossover(p1, p2)$ \\ According to Section 3.2.1
20. $p1 = mutate(p1)$, $p2 = mutate(p2)$ \\ According to Section 3.2.1
21. $P = P \cup \{p1, p2\}$
22. $P_{size} = size(P)$
23. End while
24. If $P_{size} > NP$
25. $P = P(1 : NP, :)$ \\ Population size is kept as NP
26. End if
27. $gen = gen + 1$
28. End while
29. Output the best individual

3.2.2. Simulated Annealing Algorithm

The SA is different from the GA and other swarm intelligence optimization algorithms that solve problems using the force of the swarm. After performing several neighborhood operations on a current solution, the SA finally obtains the optimal solution that it can search for. The characteristic of the SA in the search process is to accept a solution worse than the current solution with a certain probability. The SA tries to accept a new solution that is slightly worse than the local optimal solution as the current solution and then searches for the current solution. Once the SA finds a better solution than the local optimal solution, it means that the SA has escaped from the local optimal solution.

In this paper, since the value range of each gene in an individual will be different after the solution space is compressed, the traditional neighborhood search strategies, the exchange, reversal, and insertion operations, are not applicable. So, we use the mutation operator designed in Section 3.2.1 to generate new solutions. Meanwhile, the mutation probability should be correspondingly increased.

So, we propose a hybrid algorithm based on the SA algorithm and the nearest neighbor algorithm. The algorithm flow is shown in Algorithm 4.

Algorithm 4 SA-NN

1. Input: instances
2. Set initial temperature T, cooling factor α, stop iteration temperature T_{stop}
3. Solution space compression \\ According to Section 3.1.1
4. Generate an initialization solution p_0, set the current solution $p_{curr} = p_0$
5. Set a best solution $p_{best} = p_0$
6. $h(p_{best}) = fitness(F(p_0))$ \\ Objective value of individual i, according to Equations (17) and (18) and
7. \\ Algorithm 2. Fitness evaluation according to Equation (22)
8. Whereas $T > T_{stop}$
9. For $i = 1 : L$ \\ Metropolis chain length L
10. Generate a new solution p_{new} from p_{curr} based on the neighborhood search rules
11. If $h(p_{new}) > h(p_{curr})$
12. $p_{curr} = p_{new}$
13. Else
14. $r = U[0,1]$ \\ Generate a random number
15. If $r < exp[(h(p_{new}) - h(p_{curr}))/T]$
16. $p_{curr} = p_{new}$
17. End if
18. End if
19. If $h(p_{curr}) < h(p_{best})$
20. $p_{best} = p_{curr}$
21. $h(p_{best}) = h(p_{curr})$
22. End if
23. End for
24. $T = \alpha * T$
25. End while
26. Output the best individual

4. Experimental Results

In this section, first, the data sets and parameter settings are described. Then, two groups of experiments are conducted to illustrate the feasibility and superiority of the algorithms. In the first group of experiments, small-scale instances with different degrees of sparsity solution space were used to evaluate the effectiveness of the solution space compression mechanism. Then, in the second group of experiments, the performances of the GA-NN, SA-NN, brute force algorithm, BACO [25], and SJSA [23] methods were compared. In the modeled problems in [23,25], one UAV or vehicle can perform multiple tasks. In the problem modeled in this paper, there is not only a case where a UAV can perform multiple tasks but also a case where multiple UAVs can perform a task cooperatively. All the compared algorithms were implemented in MATLAB R2018b on a workstation (Intel(R) Core (TM) i7-8700 CPU @ 3.20GHz 3.19 GHz, 16.00 GB of RAM) and run independently on each instance 30 times.

In the brute force algorithm, we traversed all the task assignment schemes in the upper level. In the lower level, we used the same nearest neighbor algorithm used in the GA-NN and SA-NN methods to construct the paths.

4.1. Data Sets

The test instances included the locations of the task points, requirements of the reconnaissance tasks, abilities of the UAVs, and information about the buildings. The normal distribution function was used to generate the task requirements and UAV abilities. The locations of the reconnaissance tasks were randomly generated in the selected scene. The number of UAVs was chosen from $\{3, 4\}$. The number of tasks was chosen from $\{10, 15, 20\}$. The solution space included four different degrees of sparsity. We designed a total of 24 different instances. Here, we specified that if the number of feasible solutions in the instances was less than 0.001 times the size of the original solution space, this type of instance was a sparse type; otherwise, it was a dense type.

4.2. Parameter Settings

In order to fairly compare the running times of the two algorithms, the parameters were adjusted appropriately so that the evaluation times of the individuals were the same when the two algorithms were run once, respectively. We carried out experiments on Instances 1 and 4, respectively (the details of Instances 1 and 4 are shown in Table 3). Through the experimental results, it was found that the GA-NN and SA-NN methods had the best performance when the relevant parameters were set as follows: the population size was set to 200, and 500 generations were given as the training-stopping criterion. The crossover rate was 0.8 and the mutation rate was 0.15. The stop iteration temperature $T_{stop} = 0.001$. The cooling factor $\alpha = 0.99$. The initial temperature $T = 285$. The metropolis chain length $L = 80$.

Table 3. Verification experiment of solution space compression mechanism.

Instance	Scale	Concentration	Brute Force Algorithm		GA-NN without SSCM		GA-NN	
	$i-j-k$		t	min	t	min	t	min
1	3-10-1	Sparse (0.00002 times)	1.439 s	**2444**	**0.688 s**	2762	0.937 s	**2444**
2	3-10-2	Sparse (0.00014 times)	43.892 s	**2036**	**0.706 s**	2585	1.022 s	**2036**
3	3-10-3	Dense (0.003 times)	451.579 s	**1928**	**0.716 s**	2776	1.185 s	**1928**
4	3-10-4	Dense (0.019 times)	–	–	**0.709 s**	2762	1.209 s	**1478**

4.3. Performance Evaluation of Solution Space Compression Mechanism

In the first group of experiments, we compared the GA-NN method with the brute force algorithm to verify the accuracy of the GA-NN method and compared the GA-NN method and the GA-NN method without the SSCM to verify the effectiveness of the SSCM. In the first group of experiments, the number of UAVs was 3, the number of reconnaissance tasks was 10, and the solution space had 4 different densities.

According to theoretical analysis, the performance of the algorithm using the SSCM was better than that of the algorithm without the SSCM. According to the number of feasible solutions, the solution space of the instances was divided into sparse and dense. Therefore, when conducting the experiments on instances with sparse solution spaces, the performance of the algorithm was significantly improved after using the SSCM. Compared to sparse solution spaces, when conducting experiments on instances with dense solution spaces, the effect was not obvious.

In order to test the performance of the solution space compression mechanism, the brute force algorithm, GA-NN method without the SSCM, and GA-NN method were tested on different instances with different solution spaces. The experimental results are shown in Table 3. $i-j-k$ represents the number of UAVs, reconnaissance tasks, and sparse type, respectively. '–' represents data that cannot be calculated. 'Concentration' represents the proportion of feasible solutions to the whole solution space. '(0.00002 times)' represents that the number of feasible solutions is 0.00002 times the size of the original solution space. The minimum values of the objective value were counted because the objective function was to minimize the total tour length of the UAVs.

From the experimental data, we can draw the following conclusions:

(1) When the solution space was sparse, the brute force algorithm quickly found the optimal solution, but when the solution space became dense, the running time of the brute force algorithm became immeasurable.

(2) It was found that the optimal objective value obtained using the GA-NN method was consistent with the optimal objective value obtained using the brute force algorithm, whereas the gap between the optimal objective value obtained using the GA-NN method

without the SSCM and the real optimal objective value became larger with the gradual increase in the density of the solution space.

(3) It was found that the running time of the GA-NN method without the SSCM was faster than that of the GA-NN method alone. The reason is that during the operation of the GA-NN method without the SSCM, the newly generated solutions were not always feasible. If the solution was not feasible, the path of the UAVs was not planned so it took less time.

4.4. ARIRTP Problem-Solving Results

In the second group of experiments, we designed different scale instances with four different density solution spaces to compare the performance of the GA-NN and SA-NN methods with the brute force algorithm, BACO, and SJSA. There were a total of 24 instances. The experimental results are shown in Table 4. In order to save space, we selected 12 groups of instances to show their evolution curves. Among them, '3-10-1' represents three UAVs, ten tasks, and the solution space of the first sparse type. The routes of the UAVs in Instance 3-10-1 are shown in Figure 5.

Table 4. Experimental results.

Instance	Scale	Concentration	Brute Force Algorithm		SA-NN		GA-NN		BACO		SJSA	
	$i-j-k$		t	min	t	min	t	min	t	min	t	min
1	3-10-1	Sparse (0.00002 times)	1.439 s	**2444**	8.178 s	**2444**	0.937 s	**2444**	**0.771 s**	**2444**	9.101 s	**2444**
2	3-10-2	Sparse (0.00014 times)	43.892 s	**2036**	7.269 s	**2038**	1.022 s	**2036**	**0.836 s**	**2038**	7.026 s	**2036**
3	3-10-3	Dense (0.003 times)	451.579 s	**1928**	7.676 s	**1928**	1.185 s	**1928**	**1.089 s**	**1928**	8.755 s	**1928**
4	3-10-4	Dense (0.019 times)	–	–	7.624 s	**1478**	1.209 s	**1478**	**1.038 s**	**1478**	8.034 s	**1478**
5	3-15-1	Sparse (0.00000011 times)	321.018 s	**3620**	11.582 s	**3620**	**0.945 s**	**3620**	1.912 s	**3620**	13.135 s	**3620**
6	3-15-2	Sparse (0.00038 times)	–	–	12.310 s	**2878**	**1.144 s**	**2878**	1.384 s	**2878**	14.347 s	**2878**
7	3-15-3	Dense (0.0031 times)	–	–	11.724 s	2144	**1.135 s**	**2128**	1.209 s	**2128**	13.078 s	**2128**
8	3-15-4	Dense (0.0111 times)	–	–	8.901 s	**252.05**	**1.874 s**	**252.05**	1.921 s	**252.05**	11.668 s	**252.05**
9	3-20-1	Sparse (0.0000000006 times)	–	–	17.188 s	**4206**	**1.767 s**	**4206**	1.861 s	**4206**	20.181 s	**4206**
10	3-20-2	Sparse (0.000002 times)	–	–	15.059 s	**3434**	**1.957 s**	**3434**	2.762 s	**3434**	17.796 s	**3434**
11	3-20-3	Sparse (0.00036 times)	–	–	16.316 s	4028	**2.624 s**	**2960**	3.118 s	**2960**	17.817 s	**2960**
12	3-20-4	Dense (0.0028 times)	–	–	10.665 s	**308.8**	**2.737 s**	**308.8**	3.564 s	**308.8**	15.101 s	**308.8**
13	4-10-1	Sparse (0.00002 times)	–	–	9.791 s	**2444**	**1.245 s**	**2444**	3.570 s	**2444**	13.268 s	**2444**
14	4-10-2	Sparse (0.00089 times)	–	–	8.426 s	**2037**	**1.536 s**	**2037**	2.854 s	**2037**	14.786 s	**2037**
15	4-10-3	Dense (0.0014 times)	–	–	8.942 s	**2170**	**1.785 s**	**2170**	3.859 s	**2170**	14.135 s	**2170**
16	4-10-4	Dense (0.0601 times)	–	–	8.908 s	**1477**	**1.733 s**	**1477**	2.616 s	**1477**	12.078 s	**1477**
17	4-15-1	Sparse (0.00000012 times)	–	–	13.933 s	**3620**	**2.579 s**	**3620**	3.277 s	**3620**	15.347 s	3639
18	4-15-2	Sparse (0.0000022 times)	–	–	15.607 s	**3576**	**3.320 s**	**3576**	3.645 s	**3576**	17.455 s	3693
19	4-15-3	Sparse (0.000048 times)	–	–	13.645 s	**2730**	**3.428 s**	**2730**	3.716 s	**2730**	16.078 s	2983

Table 4. *Cont.*

Instance	Scale	Concentration	Brute Force Algorithm		SA-NN		GA-NN		BACO		SJSA
	$i-j-k$		t	min	t	min	t	min	t	min	t	min
20	4-15-4	Dense (0.1501 times)	–	–	9.734 s	**1675**	**2.721 s**	**1675**	3.233 s	**1675**	13.263 s	1836
21	4-20-1	Sparse (0.0000000007 times)	–	–	15.777 s	**3611**	**3.495 s**	4206	5.196 s	4310	19.181 s	4341
22	4-20-2	Sparse (0.0000014 times)	–	–	14.654 s	**3710**	**5.080 s**	3715	6.401 s	**3710**	19.822 s	3857
23	4-20-3	Sparse (0.0000338 times)	–	–	12.500 s	**2826**	**4.025 s**	4028	6.005 s	4152	19.796 s	3169
24	4-20-4	Dense (0.087 times)	–	–	12.305 s	**1895**	**5.868 s**	2066	7.163 s	2208	15.817 s	1930

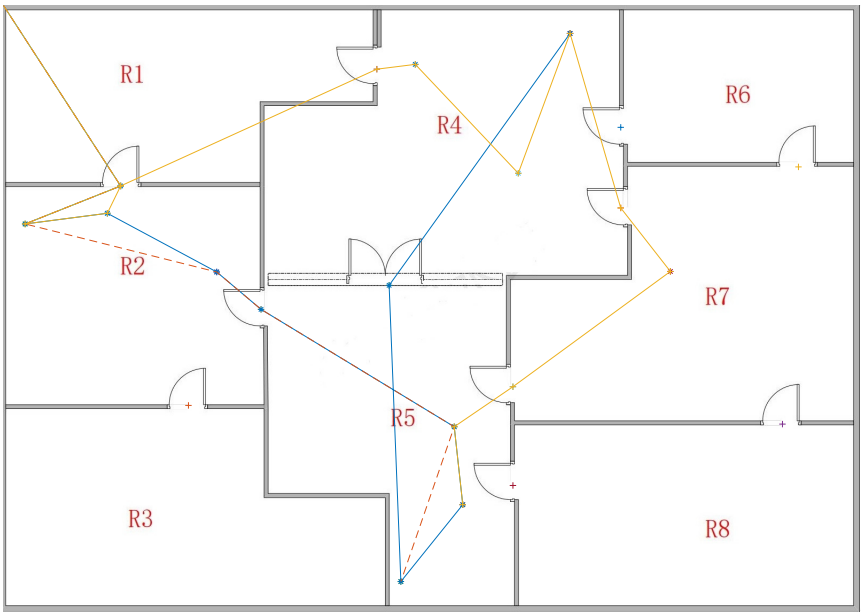

Figure 5. The routes of the UAVs in Instance 3-10-1.

From the experimental data, we can draw the following conclusions:

(1) With the increase in the number of reconnaissance tasks or the gradual density of the solution space, the brute force algorithm failed.

(2) As can be seen from the experimental data, in most instances, the objective function values obtained using the SA-NN and GA-NN methods after the convergence of the algorithms were the same, but the running time of the SA-NN method was longer than that of the GA-NN method.

(3) However, in some large-scale instances, the SA-NN method performed better than the GA-NN method.

(4) The GA-NN method was superior to the SJSA in terms of both the algorithm convergence time and optimal objective function values obtained after the algorithm convergence. After the algorithm convergence, the optimal objective function values obtained using the GA-NN method and the BACO in most instances were the same. However, in some large-scale instances, the objective function value obtained using the GA-NN method was better than that obtained using the BACO. Moreover, the GA-NN method converged faster than the BACO.

(5) The SA-NN method converged faster than the SJSA. After the algorithm convergence, in most instances, the optimal objective function values obtained using the SA-NN method and the SJSA were the same. However, in some large-scale instances, the objective function value obtained using the SA-NN method was better than that obtained using the SJSA.

4.5. Performance Evaluation of the GA-NN and SA-NN Methods

The objective value evolution curves of the GA-NN and SA-NN methods for Instances 3 and 4 are shown in Figure 6 and Figure 7, respectively. However, it can be seen from the figures that the GA-NN method converged faster than the SA-NN method. In this section, we used the Mann–Whitney U test and Kolmogorov–Smirnov test with 95% confidence to test whether there were significant differences between the GA-NN and SA-NN methods. The statistical hypothesis test results are shown in Table 5. As far as the Mann–Whitney U test and Kolmogorov–Smirnov test are concerned, the performance of the GA-NN method was no different from that of the SA-NN method. However, in some large-scale instances, the SA-NN method performed better than the GA-NN method.

Table 5. Statistical hypothesis test results.

	1	2	3	4	5	6	7	8	9	10	11	12
Mann–Whitney U test	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/28/2	0/0/30
Kolmogorov–Smirnov test	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/27/3	0/0/30
Instances	13	14	15	16	17	18	19	20	21	22	23	24
Mann–Whitney U test	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	28/0/2	27/0/3	30/0/0	24/0/6
Kolmogorov–Smirnov test	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	0/0/30	30/0/0	29/0/1	30/0/0	22/0/8

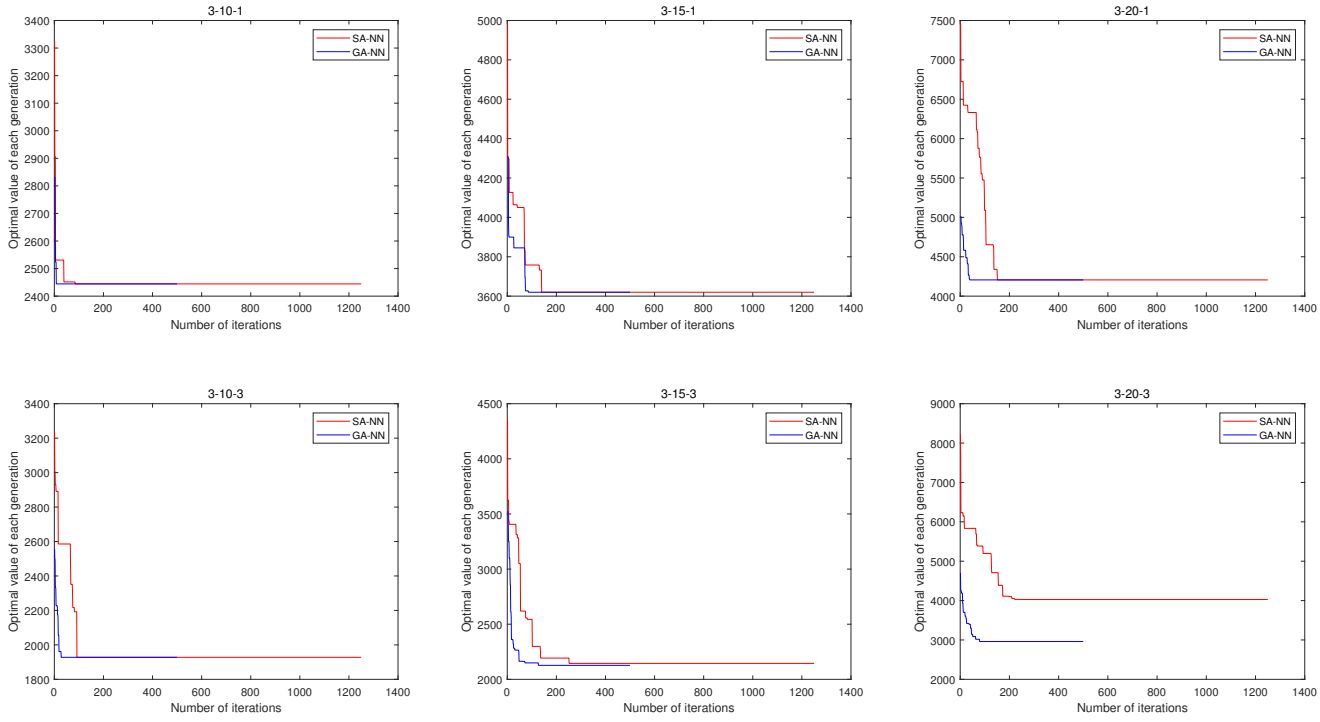

Figure 6. Objective value evolution curve for Instance 3.

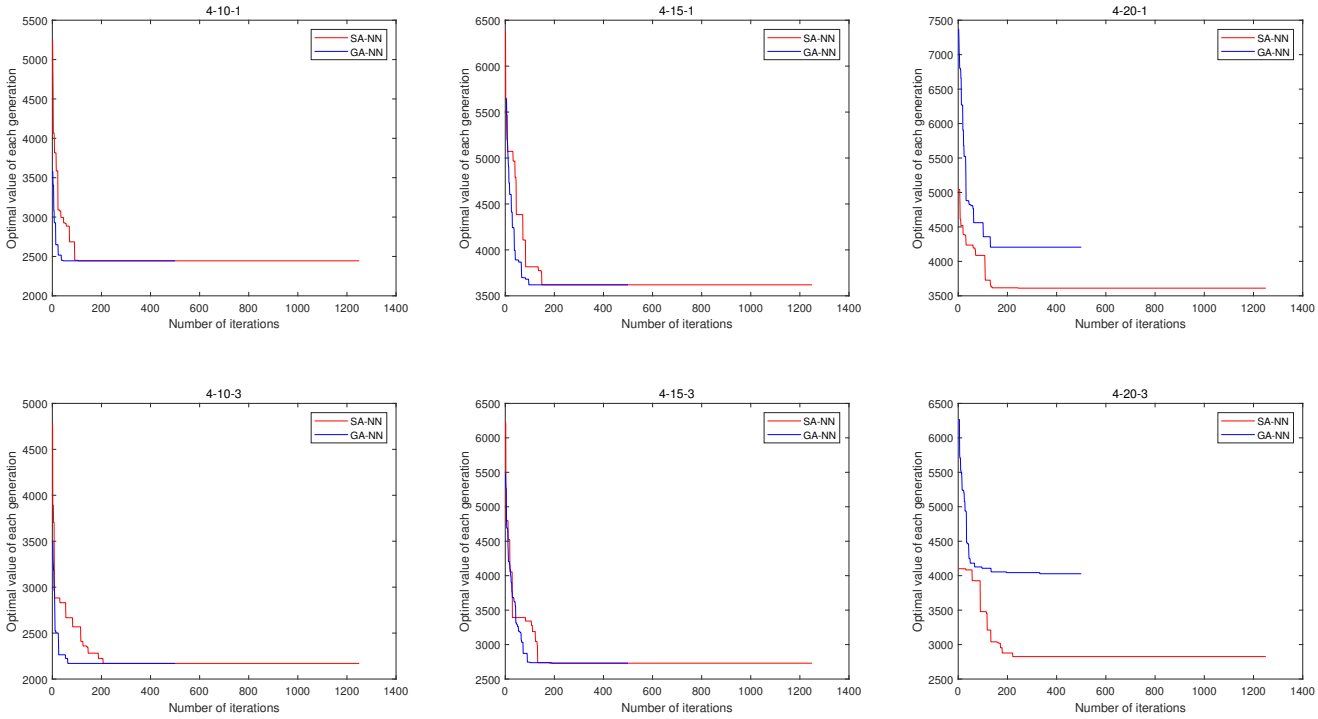

Figure 7. Objective value evolution curve for Instance 4.

Remark 2. *Instances (†/§/ ≈): †, §, and ≈ represent that the performance of the SA-NN method is superior to, inferior to, and no different from the GA-NN method, respectively.*

5. Conclusions and Perspectives

This paper focuses on the ability-restricted indoor reconnaissance task-planning problem for multiple UAVs. First, the ARIRTP problem is modeled as a combinatorial optimization problem in which the multi-dimensional requirements of the reconnaissance tasks, multi-dimensional abilities of UAVs, and impact of the connectivity between indoor rooms on the path planning of UAVs are considered. Second, a bi-level problem-solving framework is proposed. The upper level uses an iterative search algorithm to solve the task assignment problem of UAVs. According to the characteristics of the problem, a solution space compression mechanism is proposed to make the generated task assignment schemes meet the requirements of the reconnaissance tasks. The lower level uses the nearest neighbor algorithm to quickly construct the path sequence of UAVs. Third, a hybrid algorithm based on the GA and the nearest neighbor algorithm and a hybrid algorithm based on the SA and the nearest neighbor algorithm are proposed to solve the ARIRTP problem under the bi-level problem-solving framework. The experimental data show that the proposed algorithms, the GA-NN and SA-NN, can solve the ARIRTP problem quickly and accurately. The performance of the GA-NN method is no different from that of the SA-NN method; however, the GA-NN method runs slightly faster. In large-scale instances, the performance of the SA-NN method is slightly better than that of the GA-NN method. Furthermore, the GA-NN and SA-NN methods perform better than the SJSA, and the GA-NN method also performs better than the BACO.

This paper also has some limitations. We assume that the indoor environment has full coverage for communication and that UAVs can communicate with each other. However, due to the complex indoor environment and large number of walls, the communication between UAVs has a certain attenuation.

In future work, we will consider the communication between UAVs in an indoor environment. The connection relationship of the communication topology between UAV teams is helpful for them to cooperate in reconnaissance, rounding up, and other tasks. Initially, we plan to add a maximum communication distance as a constraint to ensure

smooth communication between UAV teams. Subsequently, the attenuation of the indoor environment to communication may be considered to make the communication model closer to reality. In indoor environments, when UAVs pass through a door, there are certain risks. In the future, we will consider introducing a risk coefficient to measure the risk degrees of the paths of UAVs.

Author Contributions: Conceptualization, R.Z.; methodology, B.X.; software, R.Z.; validation, R.Z.; formal analysis, R.Z.; investigation, R.Z.; resources, R.Z.; data curation, R.Z.; writing—original draft preparation, R.Z.; writing—review and editing, R.Z.; visualization, R.Z.; supervision, Y.D.; funding acquisition, Q.W. and L.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Outstanding Youth Talent Support Program 61822304, NSFC, under Grant 61873033, the Basic Science Center Programs of the NSFC under Grant 62088101, the Beijing Advanced Innovation Center for Intelligent Robots and Systems, the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100), the Shanghai Municipal Commission of Science and Technology Project (19511132101), the National Natural Science Fund of China under Grant 62003044, and the National Science Fund for Distinguished Young Scholars of China under Grant 62025301.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ARIRTP	Ability-restricted indoor reconnaissance task planning
SSCM	Solution space compression mechanism
GA	Genetic algorithm
SA	Simulated annealing algorithm
NN	Nearest neighbor algorithm
GA-NN	Genetic algorithm–nearest neighbor algorithm
SA-NN	Simulated annealing algorithm–nearest neighbor algorithm
UAVs	Unmanned aerial vehicles
SJSA	Swap-and-judge simulated annealing algorithm
BACO	Bi-level ant colony optimization
CEVRP	Capacitated electric vehicle routing problem

References

1. Allam, A.; Nemra, A.; Tadjine, M. Parametric and implicit features-based UAV-UGVs time-varying formation tracking: Dynamic approach. *Unmanned Syst.* **2022**, *10*, 109–128. [CrossRef]
2. Zammit, C.; Kampen, E.J. Comparison between A* and RRT algorithms for 3D UAV path planning. *Unmanned Syst.* **2022**, *10*, 129–146. [CrossRef]
3. Khan, S.; Tufail, M.; Khan, M.T. A novel framework for multiple ground target detection, recognition and inspection in precision agriculture applications using a UAV. *Unmanned Syst.* **2022**, *10*, 45–56. [CrossRef]
4. Li, Q.; Hua, Y.; Dong, X.; Yu, J.; Ren, Z. Time-varying formation tracking control for unmanned aerial vehicles with the leader's unknown input and obstacle avoidance: Theories and applications. *Electronics* **2022**, *11*, 2334. [CrossRef]
5. Zhang, J.; Chen, Y.; Yang, Q.; Lu, Y.; Shi, G.; Wang, S.; Hu, J. Dynamic task allocation of multiple UAVs based on improved A-QCDPSO. *Electronics* **2022**, *11*, 1028. [CrossRef]
6. Chung, S.H.; Sah, B.; Lee, J. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Comput. Oper. Res.* **2020**, *123*, 105004. [CrossRef]
7. Macrina, G.; Pugliese, L.D.; Guerriero, F.; Laporte, G. Drone-aided routing: A literature review. *Transp. Res. Part Emerg. Technol.* **2020**, *120*, 102762. [CrossRef]
8. Tamke, F.; Buscher, U. A branch-and-cut algorithm for the vehicle routing problem with drones. *Transp. Res. Part Methodol.* **2021**, *144*, 174–203. [CrossRef]
9. Liu, Y.; Liu, Z.; Shi, J.M.; Wu, G.; Pedrycz, W. Two-echelon routing problem for parcel delivery by cooperated truck and drone. *IEEE Trans. Syst. Man, Cybern. Syst.* **2021**, *51*, 7450–7465. [CrossRef]
10. Vasquez, S.A.; Angulo, G.; Klapp, M.A. An exact solution method for the TSP with drone based on decomposition. *Comput. Oper. Res.* **2021**, *127*, 105127. [CrossRef]

11. Luo, Z.; Pan, B.; Zhang, Z.; Liu, Z.; Lim, A. The multi-visit traveling salesman problem with multi-drones. *Transp. Res. Part Emerg. Technol.* **2021**, *128*, 103172. [CrossRef]
12. Cui, Y.J.; Dong, W.H.; Hu, D.X.; Liu, H.B. The application of improved harmony search algorithm to multi-UAV task assignment. *Electronics* **2022**, *11*, 1171. [CrossRef]
13. Khan, A.; Zhang, J.; Ahmad, S.; Memon, S.; Qureshi, H.A.; Ishfaq, M. Dynamic positioning and energy-efficient path planning for disaster scenarios in 5G-assisted multi-UAV environments. *Electronics* **2022**, *11*, 2197. [CrossRef]
14. Rudys, S.; Ragulis, P.; Laučys, A.; Bručas, D.; Pomarnacki, R.; Plonis, D. Investigation of UAV detection by different solid-state marine radars. *Electronics* **2022**, *11*, 2502. [CrossRef]
15. Mostafa, S.A.; Mustapha, A.; Gunasekaran, S.S.; Ahmad, M.S.; Mohammed, M.A.; Parwekar, P.; Kadry, S. An agent architecture for autonomous UAV flight control in object classification and recognition missions. *Soft Comput.* **2021**. [CrossRef]
16. Greenwood, W.W.; Lynch, J.P.; Zekkos, D. Applications of UAVs in civil infrastructure. *J. Infrastruct. Syst.* **2019**, *25*, 04019002. [CrossRef]
17. Sun, F.; Wang, X.; Zhang, R. Task scheduling system for UAV operations in agricultural plant protection environment. *J. Ambient. Intell. Humaniz. Comput.* **2020**. [CrossRef]
18. Shi, T.; Wang, H.; Cui, W.; Ren, L. Indoor space target searching based on EEG and EOG for UAV. *Soft Comput.* **2019**, *23*, 11199–11215. [CrossRef]
19. Sampedro, C.; Rodriguez-Ramos, A.; Bavle, H.; Carrio, A.; de la Puente, P.; Campoy, P. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *J. Intell. Robot. Syst.* **2019**, *95*, 601–627. [CrossRef]
20. Khosiawan, Y.; Park, Y.; Moon, I.; Nilakantan, J.M.; Nielsen, I. Task scheduling system for UAV operations in indoor environment. *Neural Comput. Appl.* **2019**, *31*, 5431–5459. [CrossRef]
21. González de Santos, L.M.; Frías Nores, E.; Martínez Sánchez, J.; González Jorge, H. Indoor path-planning algorithm for UAV-based contact inspection. *Sensors* **2021**, *21*, 642. [CrossRef] [PubMed]
22. Bouzid, Y.; Bestaoui, Y.; Siguerdidjane, H. Guidance-control system of a quadrotor for optimal coverage in cluttered environment with a limited onboard energy. *J. Intell. Robot. Syst.* **2019**, *95*, 707–730. [CrossRef]
23. Huo, L.; Zhu, J.; Wu, G.; Li, Z. A novel simulated annealing based strategy for balanced UAV task assignment and path planning. *Sensors* **2020**, *20*, 4769. [CrossRef] [PubMed]
24. Chen, H.X.; Nan, Y.; Yang, Y. Multi-UAV reconnaissance task assignment for heterogeneous targets based on modified symbiotic organisms search algorithm. *Sensors* **2019**, *19*, 734. [CrossRef]
25. Jia, Y.H.; Mei, Y.; Zhang, M.J. A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem. *IEEE Trans. Cybern.* **2022**, *52*, 10855–10868. [CrossRef]
26. Gomez, C.; Fehr, M.; Millane, A.; Hernandez, A.C.; Nieto, J.; Barber, R.; Siegwart, R. Hybrid topological and 3D dense mapping through autonomous exploration for large indoor environments. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May 2020–31 August 2020; pp. 9673–9679.
27. Ruan, W.Y.; Duan, H.B. Multi-UAV obstacle avoidance control via multi-objective social learning pigeon-inspired optimization. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 740–748. [CrossRef]
28. Dai, H.; Lu, W.; Li, X.; Yang, J.; Meng, D.; Liu, Y.; Liang, B. Cooperative planning of multi-agent systems based on task-oriented knowledge fusion with graph neural networks. *Front. Inf. Technol. Electron. Eng.* **2022**, *23*, 1069–1076. [CrossRef]
29. Dhiman, N.K.; Deodhare, D.; Khemani, D. Where am I? Creating spatial awareness in unmanned ground robots using SLAM: A survey. *Sadhana* **2015**, *40*, 1385–1433. [CrossRef]
30. Mccammon, S.; Hollinger, G.A. Topological path planning for autonomous information gathering. *Auton. Robot.* **2021**, *45*, 821–842. [CrossRef]
31. He, S.W.; Kilgour, D.M.; Hipel, K.W. A three-level hierarchical graph model for conflict resolution. *IEEE Trans. Syst. Man, Cybern. Syst.* **2021**, *51*, 1424–1433. [CrossRef]
32. Zhu, Z.M.; Kilgour, D.M.; Hipel, K.W. A new approach to coalition analysis within the graph model. *IEEE Trans. Syst. Man, Cybern. Syst.* **2020**, *50*, 2231–2241. [CrossRef]
33. Zhang, H.; Xin, B.; Dou, Li.; Chen, J.; Hirota, K. A review of cooperative path planning of an unmanned aerial vehicle group. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 1671–1694. [CrossRef]
34. Ding, Y.L.; Xin, B.; Chen, J. A review of recent advances in coordination between unmanned aerial and ground vehicles. *Unmanned Syst.* **2021**, *09*, 97–117. [CrossRef]
35. Chen, J.; Ding, Y.; Xin, B.; Yang, Q.; Fang, H. A unifying framework for human-agent collaborative systems-part I: Element and relation analysis. *IEEE Trans. Cybern.* **2022**, *52*, 138–151. [CrossRef]
36. Chen, J.; Ding, Y.; Xin, B.; Yang, Q.; Fang, H. A unifying framework for human-agent collaborative systems-part II: Design procedure and application. *IEEE Trans. Cybern.* **2021**, *52*, 11990–12002.
37. Tao, Y.; Wen, Y.; Gao, H.; Wang, T.; Wan, J.; Lan, J. A path-planning method for wall surface inspection robot based on improved genetic algorithm. *Electronics* **2022**, *11*, 1192. [CrossRef]
38. Xiao, S.; Tan, X.; Wang, J. A simulated annealing algorithm and grid map-based UAV coverage path planning method for 3D reconstruction. *Electronics* **2021**, *10*, 853. [CrossRef]
39. Available online: https://mavrovouniotis.github.io/EVRPcompetition2020/ (accessed on 19 July 2020).

40. Wang, Y.Y.; Jiao, X.H. Multi-objective energy management for PHEV using Pontryagin's minimum principle and particle swarm optimization online. *Sci.-China-Inf. Sci.* **2021**, *64*, 119204. [CrossRef]
41. Chen, L.; Xin, B.; Chen, J. Interactive multiobjective evolutionary algorithm based on decomposition and compression. *Sci.-China-Inf. Sci.* **2021**, *64*, 202201. [CrossRef]