# Decentralized Multi-agent Task Planning for Heterogeneous UAV Swarm

*JIA Tao*[*], *XU Haihang*, *YAN Hongtao*, *DU Junjie*

Aerospace Technology Research Institute, China Aerodynamics Research and Development Center, Mianyang 621000, P. R. China

(Received 18 June 2020; revised 17 July 2020; accepted 25 July 2020)

**Abstract:** A decentralized task planning algorithm is proposed for heterogeneous unmanned aerial vehicle (UAV) swarm with different capabilities. The algorithm extends the consensus-based bundle algorithm (CBBA) to account for a more realistic and complex environment. The extension of the algorithm includes handling multi-agent task that requires multiple UAVs collaboratively completed in coordination, and consideration of avoiding obstacles in task scenarios. We propose a new consensus algorithm to solve the multi-agent task allocation problem and use the Dubins algorithm to design feasible paths for UAVs to avoid obstacles and consider motion constraints. Experimental results show that the CBBA extension algorithm can converge to a conflict-free and feasible solution for multi-agent task planning problems.

**Key words:** task allocation; unmanned aerial vehicle (UAV) swarm; consensus-based bundle algorithm (CBBA); multi-agent task; obstacle avoidance

**CLC number:** V279 **Document code:** A **Article ID:** 1005-1120(2020)04-0528-11

## 0 Introduction

With the development of unmanned aerial vehicle (UAV) technology and its corresponding supporting technologies, the mission capability of UAV swarm has been expanding and has gradually been transformed from an early reconnaissance and monitoring platform into a multi-functional multi-purpose platform[1]. Due to the limited capabilities of individual types of UAV, it is not possible to use a UAV to accomplish all tasks. By selecting a variety of heterogeneous UAVs to form a swarm and accomplishing tasks together, we can solve the capability limitation of UAV and improve the efficiency of performing tasks. And this brings up another important issue, which is to ensure the coordination and cooperation of the UAV swarm. This is critical for a swarm to be able to complete complex tasks efficiently and successfully. Therefore, it is very important to develop effective algorithms to solve the task allocation problem within the swarm[2-4].

Task allocation problem (TAP), assigns a limited number of agents to a limited number of tasks while satisfying certain constraints, and gets a higher task reward as much as possible. The solution to this problem is divided into centralized and decentralized methods. More and more researchers are paying attention to decentralized methods, which are more suitable for the real world[5]. The commonly used decentralized solution methods mainly include market auction methods based on contract nets, distributed Markov decision methods, decentralized model predictive control methods, dynamic decentralized constrained optimization methods and so on[6-8]. Among them, the auction algorithm is oriented to the dynamic execution and autonomous solution of the task, and considers the information interaction and negotiation between the agents in the solution process, which is a relatively easy to implement distributed algorithm[9-11]. The consensus-

*Corresponding author, E-mail address: Tobejiatao@163.com.

based bundle algorithm (CBBA) is a multi-task allocation algorithm that utilizes auction algorithm[12]. It allows agent to bid on each task, and then develops a consensus algorithm to resolve allocation conflicts between agents.

By designing complex tasks with more constraints, we can extend the TAP problem to bring it closer to reality. In the real world, some complex tasks require multiple agents to work together to complete, which is classified as a multi-agent task. Distributed greedy algorithm is used to solve the multi-agent task assignment problem, where a group of agents need to select tasks from their admissible task sets[13]. The objective is to find an assignment profile that maximizes the global utility. The multi-agent task assignment problem is impossible to be solved by CBBA algorithms[14]. So the consensus algorithms need to be developed to handle these more rigorous task choices and higher collaborative decision making.

Besides, the CBBA algorithm does not consider the effects of complex environmental constraints. In the real world, UAVs will inevitably encounter unreachable area such as obstacle areas and no-fly areas during the execution of missions. When the task allocation model calculates the cost of each agent to complete the task, the existence of these areas must be considered. Ref.[15] analyzed and compared two frameworks (Compromise view model and the nearest-neighbour search model) for co-operative path planning combined with task assignment of a multi-agent system in dynamic environments, and the particle swarm optimization-based method combined with the obstacle avoidance strategy was applied for path planning. The solution proposed here takes these situations into the new cost calculation function and uses the Dubins algorithm to design a feasible path for UAVs to avoid the obstacle area.

This paper first describes the model and principle of the task allocation problem and the CBBA algorithm, and then extends the CBBA algorithm to solve multi-agent tasks and develops a new consensus algorithm. Considering the obstacle avoidance problem in complex environment, the Dubins algorithm is used to design the feasible path that considers UAVs' motion constraints. Finally, a number of comparative simulation experiments are carried out with the original CBBA algorithm, and result proves that the algorithm extended in this paper can converge to a conflict-free and feasible solution which previous algorithms are unable to account for.

## 1 Background

CBBA is a distributed auction algorithm that provides a reliable approximation solution for multi-agent multi-task allocation problems. It has an iteration between two distinct phases: A bundled building phase in which each agent generates a local ordered task bundle, and a consensus phase in which the allocation conflicts are resolved by communication between neighboring agents.

### 1.1 Phase 1: bundle construction

In the first phase, the agent locally builds a bundle containing all the tasks it plans to complete and update during the distribution process. Each agent continually adds tasks to its bundle until it cannot add any other tasks. The agent contains two task lists: bundle $\boldsymbol{b}_i$ and path $\boldsymbol{p}_i$. $\boldsymbol{b}_i$ contains all the tasks that the agent $i$ will complete and be grouped in the order of the added tasks. While $\boldsymbol{p}_i$ contains the ordered sequence of tasks that agent $i$ will complete. $S_i^{p_i}$ is used as a total reward for agent $i$ to perform for the tasks contained in $\boldsymbol{p}_i$, where $S_i^{p_i \oplus_n \{j\}}$ is the total reward corresponding to the insertion of the task $j$ into the position $n$ of the path $\boldsymbol{p}_i$. Adding task $j$ to bundle $\boldsymbol{b}_i$ will result in a higher score $c_{ij}[\boldsymbol{b}_i]$.

$$c_{ij}[\boldsymbol{b}_i]=\begin{cases}0 & j\in \boldsymbol{b}_i \\ \max\limits_{n\leqslant |p_i|} S_i^{p_i \oplus_n \{j\}} - S_i^{p_i} & \text{Otherwise}\end{cases} \tag{1}$$

where $|p_i|$ is the cardinality of the list $\boldsymbol{p}_i$, and $\oplus_n$ the operation of inserting the second list after the $n$th element of the first list.

Insert a new task at all possible locations in the current path and find the highest increase in rewards. Each agent has five vectors: winning bid list $\boldsymbol{y}_i$, winning agent list $\boldsymbol{z}_i$, agent update time $\boldsymbol{s}_i$, bundle $\boldsymbol{b}_i$ and corresponding path $\boldsymbol{p}_i$. The winning agent list $\boldsymbol{z}_i$ is the agent that wins in the current bid of each task, such that, when $\boldsymbol{z}_{ij} = k$, the agent $i$ believes that the task $j$ is assigned to the agent $k$. The agent needs to know not only if the task it chooses is outbid, but also who each task is assigned to. Therefore more complex conflict resolution rules are needed to achieve better allocation.

### 1.2 Phase 2: conflict resolution

The consensus phase of CBBA is to avoid too many agents bidding for the same task. If one agent is outbid for a task, the scores for all subsequent tasks will no longer be valid. Therefore, when an agent bid is outbid, it must release all tasks added after the outbid task. When agent $i$ receives messages from another agent $k$, $\boldsymbol{z}_i$ and $\boldsymbol{s}_i$ are used to decide which agent's information is up to date for each task. Agent $i$ can take the following three possible actions on task $j$:

(1) Update: $\boldsymbol{y}_{ij}=\boldsymbol{y}_{kj}$, $\boldsymbol{z}_{ij}=\boldsymbol{z}_{kj}$;

(2) Reset: $\boldsymbol{y}_{ij}=0$, $\boldsymbol{z}_{ij}=\varnothing$;

(3) Leave: $\boldsymbol{y}_{ij}=\boldsymbol{y}_{ij}$, $\boldsymbol{z}_{ij}=\boldsymbol{z}_{ij}$.

If the decision rule changes the bid, each agent checks that if the updated or reset task is in its bundle, and if so, releases the task and all the tasks that are added to the bundle after it.

$$\begin{aligned} &\boldsymbol{y}_{i,b_{in}}=0, \boldsymbol{z}_{i,b_{in}}=\varnothing \quad && \forall n>\bar{n}_i \\ &\boldsymbol{b}_{in}=\varnothing && n>\bar{n}_i \end{aligned} \tag{2}$$

where $\boldsymbol{b}_{in}$ denotes the $n$th entry of bundle $\boldsymbol{b}_i$, and $\bar{n}_i=\min\{n: z_{i,b_{in}}\neq i\}$.

It should be noted that the winning bid and the winning agent of the task added after $\boldsymbol{b}_{in}$ are reset, because removing the bin can change the score of all subsequent tasks. From here on, the algorithm returns to the first phase and adds a new task. And the agent iterates through the two phases until they converge on a conflict-free solution.

## 2 CBBA with Multi-agent Task

The extension of CBBA proposed in this section will solve the multi-agent task problem, but keep the agent independent, allowing them to freely form groups to complete multi-agent tasks. In the bundle construction phase of the algorithm, the method of constructing task bundle $\boldsymbol{b}_i$ and path $\boldsymbol{p}_i$ is the same as the original CBBA algorithm, but the cost calculation method is different, which will be introduced in the next section. In the conflict resolution phase of the algorithm, the agent receives data from nearby agents about all task allocation information, and then uses a consensus algorithm to agree on the assignment of all tasks. This section proposes a new consensus algorithm for multi-agent tasks.

First of all, we need to determine the data that are communicated between the agents needed for the consensus algorithm. Task and agent information is stored locally when start to use CBBA. Each agent stores two vectors of length $N_m$ ($N_m$ is the number of tasks in the algorithm), the winning bid list $\boldsymbol{y}_i$ and the winning agent list $\boldsymbol{z}_i$. Each agent can use $\boldsymbol{z}_i$ to determine who has the highest bid for each task and use $\boldsymbol{y}_i$ to determine the highest bid. Problems occur when using the original CBBA consensus algorithm to process data for multi-agent tasks. Different tasks can be assigned different numbers of agents, and for tasks that require multiple assignments, the vector cannot store each allocated data. Therefore, we need to change the way storing these values in order to solve the multi-agent task allocation problem. We must convert the two vectors into a matrix to determine multiple winners and their bids.

We can combine the two vectors into a single matrix $\boldsymbol{B}$, which contains all the winning information. The matrix uses rows to display tasks and columns to display agents, so $\boldsymbol{B}_{ij}$ corresponds to the bid made by agent $i$ for task $j$, or equals 0 if the agent has not yet bid. In addition, the matrix $\boldsymbol{B}$ has size of $N_n\times N_m$, where $N_n$ is the number of agents in the simulation, and the number of non-zero values in each row should never exceed the number of agents $L_j$ required for the task. Besides we use $\boldsymbol{B}^i_{mj}$ to distinguish the local data of each agent, where $\boldsymbol{B}^i_{mj}>0$ means that agent $i$ believes that task $j$ is assigned to agent $m$. Algorithm 1 shows how to convert the vector $\boldsymbol{z}_i$, $\boldsymbol{y}_i$ into the matrix set $\boldsymbol{B}^i_{mj}$.

**Algorithm 1** Constructing the matrix set for agent $i$

```
(1)  for $j=1$ to $N_m$ do
(2)    for $m=1$ to $N_n$ do
(3)      if $\boldsymbol{z}(i,j)=m$ then
(4)        $\boldsymbol{B}^i_{mj}=\boldsymbol{y}(m,j)$
(5)      end
(6)    end
(7)  end
```

The original CBBA algorithm uses a lookup table to determine whether to update or reset the receiver's information based on the sender's information. In the case of multi-agent tasks, the receiver does not necessarily leave or update data different from himself, but is more likely to merge it, resulting in the data of both agents being corrected and saved. The multi-agent consensus algorithm is divided into two phases so that the data of the agents can be better combined to cope with the diversity of tasks.

The first phase is to compare the time information of the agent with all the data it receives and accept the newer data. By comparing the timestamp information of the agent, we can know which agent's data are newer. For example, at lines 4—8 in Algorithm 2, $\boldsymbol{s}_{km}>\boldsymbol{s}_{im}$ indicate that $k$ has more up-to-date communication data, which may be higher bid or more up-to-date allocation information and should be saved.

In the second phase, based on the information that all senders and receivers are currently up-to-date, the receiver's data are updated based on the sender's data. We first check every agent that the sender $k$ thinks to be assigned to each task $j$ at lines 10—11 in Algorithm 2. If the receiver agent $i$ believes that the task has not been assigned to the agent $m(\boldsymbol{B}^i_{mj}=0)$ and the number of agents assigned to task $j$ has not yet reached the number of requests of the task, the receiver $i$ can directly update the assignment matrix by $\boldsymbol{B}^i_{mj}=\boldsymbol{B}^k_{mj}$.

In addition, when the assignment of task $j$ is full or there is a better bid, we should update the allocation matrix of task $j$ at lines 12—14 in Algorithm 2. When the receiver $i$ thinks that the assignment of task $j$ is full, we first find the agent with the lowest bid among the agents assigned to task $j$. If the sender $k$ thinks there is a higher bid than this value, we should replace the minimum bid with the sender's data, and update the new assignment matrix with $\boldsymbol{B}^i_{nj}=0$, $\boldsymbol{B}^i_{mj}=\boldsymbol{B}^k_{mj}$ ($n$ is the lowest bid agent). There may also be a problem here: When the minimum bid is equal to the sender's data, there is a possibility of a deadlock. So we make a simple rule that when the agents' bids are equal, the agent with the higher ID has priority at lines 15—19 in Algorithm 2.

**Algorithm 2** Conflict resolution for agent $i$

```
(1)  receive $B^k_{ij}$ and $s_k$ from agent $k$
(2)  for $j = 1$ to $N_m$ do
(3)    for $m = 1$ to $N_n$ do
(4)      if $B^i_{mj}>0$ and $m \neq i$ then
(5)        if $s_{km}>s_{km}$ or $m=k$ then
(6)          $B^i_{mj} = B^k_{mj}$
(7)        end
(8)      end
(9)      if $m \neq i$ and $B^i_{mj} = 0$ and $B^k_{mj} > 0$ then
(10)       if $(B^i_{nj} > 0) < L_j$, $\forall n$ then
(11)         $B^i_{mj} = B^k_{mj}$
(12)       else if $\min(B^i_{nj})<B^k_{mj}$, $\forall n$ then
(13)         $B^i_{nj} = 0$
(14)         $B^i_{mj} = B^k_{mj}$
(15)       else if $\min(B^i_{nj})=B^k_{mj}$, $\forall n$ then
(16)         if $m>n$ then
(17)           $B^i_{nj} = 0$
(18)           $B^i_{mj} = B^k_{mj}$
(19)         end
(20)       end
(21)     end
(22)   end
(23) end
```

After checking each task and each agent assigned to it according to the above algorithm, the iteration is repeated until each agent has already become a sender and receiver. Because the winning agent and the winning bid value data are used in the bundle construction phase of the next cycle, we need to convert the matrix set $\boldsymbol{B}$ into $\boldsymbol{z}_i$ and $\boldsymbol{y}_i$ of each agent in Algorithm 3. When the agent $i$ believes that the task $j$ is assigned to himself, let $\boldsymbol{z}(i,j)$ be equal to $i$, otherwise it is equal to the agent who can get the highest reward for task $j$, and $\boldsymbol{y}(i,j)$ is the highest reward value. Although these vectors do not contain all the allocation information, they are sufficient for each agent's own allocation information.

**Algorithm 3** Restore the vector for agent $i$

```
(1)  for $j=1$ to $N_m$ do
(2)    if $B^i_{ij} > 0$ then
(3)      $z(i,j) = i$
(4)    else
(5)      $z(i,j) = \mathrm{argmax}_n(B^i_{nj})$, $\forall n$
(6)    end
(7)    $y(i,j)=\max(B^i_{nj})$, $\forall n$
(8)  end
```

## 3 CBBA with Obstacle Avoidance

The original CBBA algorithm did not mention how to calculate the distance cost between tasks. The conventional approach is to use the linear distance between the tasks without considering the mobility of the UAVs and the obstacles in the environment. In this section, a new scoring function is proposed, which uses the Dubins algorithm to design a feasible path for UAVs to avoid the obstacle area.

Dubins theory is mainly used to solve the shortest path with curvature constraints[16]. At present, there are many research results using Dubins algorithm to plan obstacle avoidance path. The construction method of Dubins set of circle-line-circle (CLC) paths using the principles of Euclidean has been introduced in detail[17], which is also cited in this section.

By selecting one of the common tangent lines of the two circles, the Dubins path can be obtained, where the starting and ending positions are on the arc, and the radius of the arc is the radius of curvature, which is determined by the turning radius of UAV. The problem is then reduced to find the common tangent of the two arcs. As shown in Fig.1, the interconnected arcs and lines form the Dubins path. If the starting point has no starting direction requirement or agents have no turning radius constraint, the starting circle can be considered as a point.

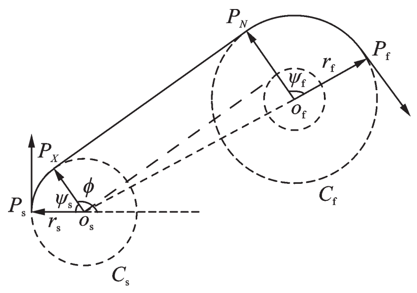


Fig.1 Dubins path with tangent

Given the starting attitude point $P_s(x_s, y_s, \phi_s)$ and ending attitude point $P_f(x_f, y_f, \phi_f)$, the starting circle $C_s$ center $o_s(x_{cs}, y_{cs})$ and the ending circle $C_f$ center $o_f(x_{cf}, y_{cf})$ can be determined firstly by

$$\begin{aligned} x_{cs} &= x_s - r_s\cos(\phi_s \pm \pi/2) \\ y_{cs} &= y_s - r_s\sin(\phi_s \pm \pi/2) \\ x_{cf} &= x_f - r_f\cos(\phi_f \pm \pi/2) \\ y_{cf} &= y_f - r_f\sin(\phi_f \pm \pi/2) \end{aligned} \tag{3}$$

where the sign depends on whether the direction of motion of the agent is clockwise or counterclockwise. Here clockwise is positive and counterclockwise is negative.

Next the position of cut-out point $P_X$ on $C_s$ and the cut-in point $P_N$ on $C_f$ can be calculated by

$$\begin{aligned} x_{P_X} &= x_{cs} + r_s\cos\phi \\ y_{P_X} &= y_{cs} + r_s\sin\phi \\ x_{P_N} &= x_{cf} + r_f\cos\phi \\ y_{P_N} &= y_{cf} + r_f\sin\phi \end{aligned} \tag{4}$$

where $\phi$ is shown in Table 1.

**Table 1 Definition of $\phi$ value**

| Direction | $\phi$ |
|---|---|
| Clockwise | $\arcsin\left(\frac{r_f - r_s}{\sqrt{(x_{cs}-x_{cf})^2+(y_{cs}-y_{cf})^2}}\right) + \arctan\left(\frac{y_{cf}-y_{cs}}{x_{cf}-x_{cs}}\right) + \frac{\pi}{2}$ |
| Counterclockwise | $-\arcsin\left(\frac{r_f - r_s}{\sqrt{(x_{cs}-x_{cf})^2+(y_{cs}-y_{cf})^2}}\right) + \arctan\left(\frac{y_{cf}-y_{cs}}{x_{cf}-x_{cs}}\right) + \frac{3\pi}{2}$ |

Finally we can get the length of the entire Dubins path, shown as

$$L_{\text{Dubins}} = r_s\psi_s + \sqrt{(x_{P_X} - x_{P_N})^2 + (y_{P_X} - y_{P_N})^2} + r_f\psi_f \tag{5}$$

where $\psi_s = \mathrm{mod}\left(2\pi + \phi - \left(\phi_s \pm \frac{\pi}{2}\right), 2\pi\right)$ and $\psi_f = \mathrm{mod}\left(2\pi + \phi_s \pm \frac{\pi}{2} - \phi, 2\pi\right)$.

When the path encounters an obstacle in the environment, we can still use the Dubins algorithm to avoid the obstacle, as shown in Fig.2. According to the different directions of motion on the obstacle circle, two obstacle avoidance paths are generated respectively. By selecting the shorter one, we can obtain a feasible path that can avoid obstacles.

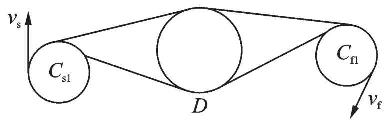


Fig.2　Dubins path with obstacle avoidance

## 4　Performance Analysis

### 4.1　Test scenario

The simulation scenario used to test the above algorithm includes two heterogeneous UAVs, Search-UAV (S-UAV) and Rescue-UAV (R-UAV), which are responsible for completing two tasks (search and rescue tasks). S-UAV is responsible for the search task and R-UAV for the rescue mission. The number of UAVs required for each task can be defined individually. Each task has 300 s time windows, a random start time, and 5 or 15 s task execution time. Each UAV has its own speed and specific fuel consumption. The initial locations of the task and UAV are randomly initialized in the task space. The following is the algorithm performance of CBBA with multi-agent tasks considering obstacle avoidance algorithms in this scenario.

### 4.2　Performance of CBBA with multi-agent tasks

To compare the performance difference between the original CBBA algorithm with the single- and multi-agent task, we will set up the following experiments: Each test contains 20 tasks, half of which are search tasks and half the rescue tasks. The first experiment verifies the original CBBA algorithm: Each task only needs one agent to complete. The second experiment is the multi-agent experiment requiring two agents for each task. The third experiment is set to a mixed experiment, in which the search task requires two S-UAVs and the rescue mission requires one R-UAV. The goal of each experiment is to maximize the sum of rewards for all UAVs completing tasks. The score function for each UAV is the reward for completing the task minus the distance penalty. Multi-agent tasks will reward each agent that completes the task, indicating the difficulty and importance of such tasks. We gradually increase the number of heterogeneous UAVs to test algorithm performance. Each experiment runs 100 times and the average data are recorded.

Fig.3(a) shows the total score of the three experiments as the number of agents increases. It can be found that at the beginning of the experiment, due to the insufficient number of heterogeneous UAVs, the scores of multi-agent tasks are lower than that of single-agent tasks, but as the number of agents increases, the scores of multi-agent tasks are significantly higher than single-agent tasks. Fig.3(b) shows that the calculation time will only increase with the number of agents, and will not change significantly due to the type of experimental

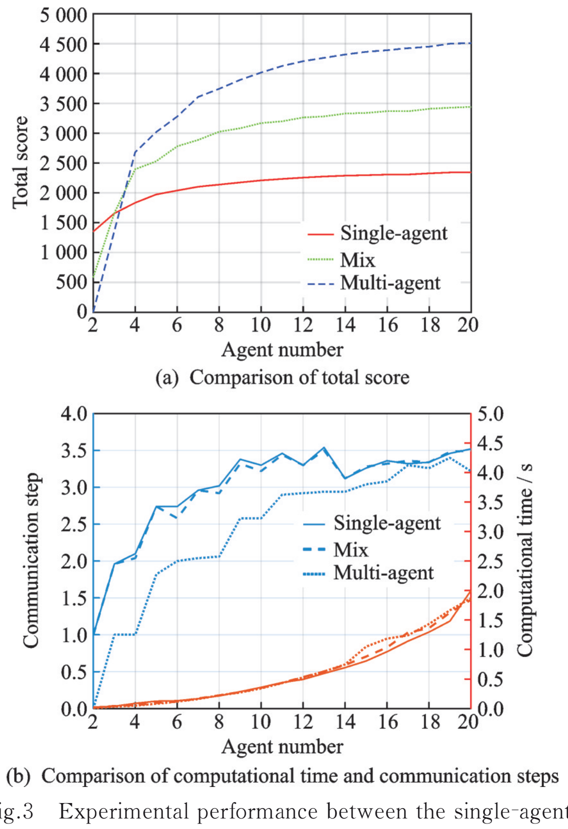


Fig.3　Experimental performance between the single-agent, multi-agent and mix experiments

tasks. It shows that the new consensus algorithm does not cause an increase in the amount of calculation.

It is also worth noting that multi-agent tasks have fewer communication steps than single-agent tasks, which can be explained by the allocation details of the agents in Fig.4 and Fig.5. Here, A means agent and T meas task. Fig.4 shows the details of a successful allocation of the single-agent task of the first experiment. In Fig.4(a), for a more intuitive display we only show the change of the $X$-axis position over time, and Fig.4 (b) shows the task sequence assigned to each UAV. Fig.5 shows a successful allocation detail for the second experiment. Compared with the single-agent task experiment that each task needs to select the optimal agent, when multiple agents form a team to complete the first task in a multi-agent task, they usually go together to perform the next most recent task, which results in less communication and distance costs. However, there will also be a change in team

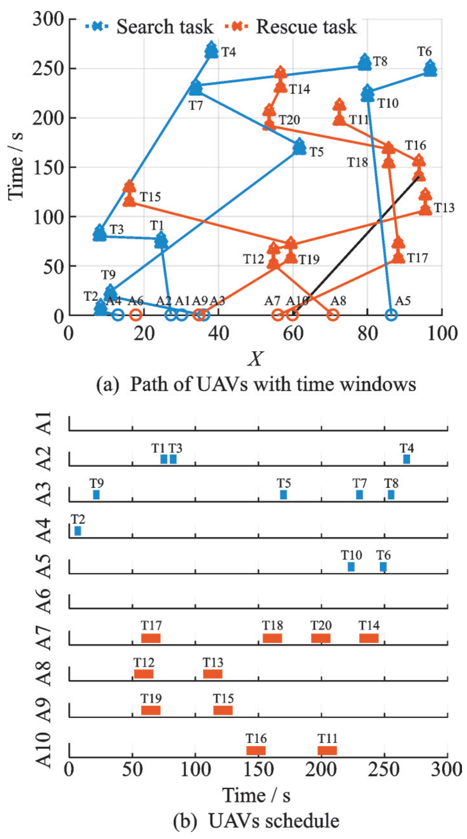


Fig.4 Distribution details of single-agent experiment (Total ten UAVs, five S-UAVs and five R-UAVs)

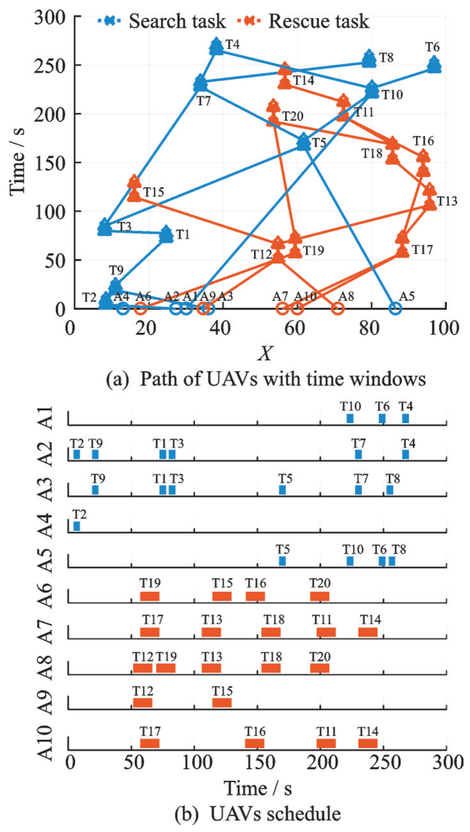


Fig.5 Distribution details of multi-agent experiment (Total ten UAVs, five S-UAVs and five R-UAVs)

members, and the closer agents will form a new team to complete the task through consensus algorithms. But in general, this phenomenon has reduced the distribution conflict between agents, thus reducing the number of communications steps.

Next we test the impact of the multi-functional UAV on the experiment. For the second experiment, we can remove the S-UAV only to complete the search task limit, so that it can complete any task. Fig.6 shows the effect of the multi-function UAV on the total score, the number of communications and the time of operation. Since the rescue task can be completed with two kinds of UAVs, it can have a better choice than the single-function UAV, so the total score is higher than that of the single-function UAV. However, this also causes an increase in the complexity of the distribution because of requiring more communication time to achieve consistency among UAVs, so it takes longer to calculate. Fig.7 shows a successful allocation detail for multi-functional experiment.

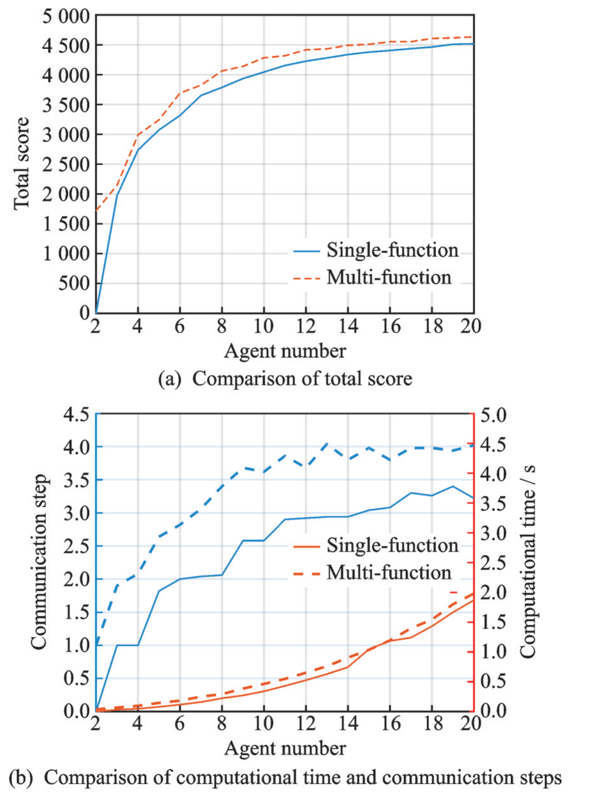


Fig.6 Agent ability effect on total score, computational time and communication steps

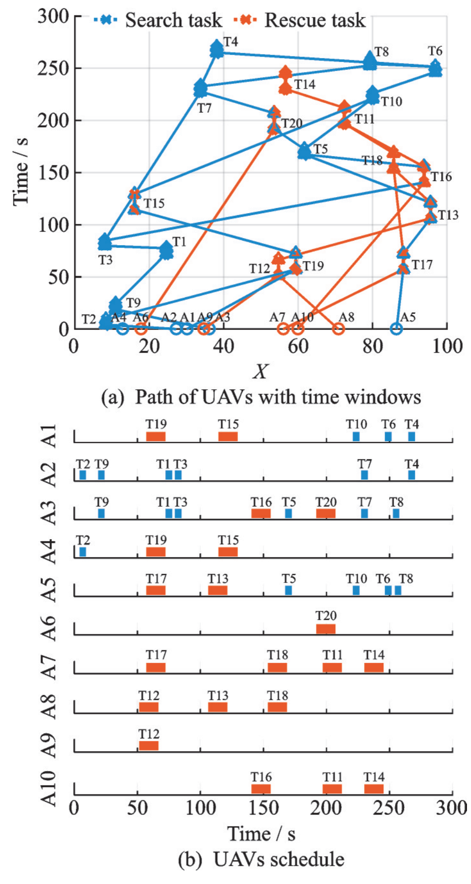


Fig.7 Distribution details of multi-functional experiment (Total ten UAVs, five S-UAVs and five R-UAVs)

## 4.3 Performance of CBBA with obstacle avoidance

In this section we verify the performance of CBBA algorithms with obstacle avoidance proposed in Section 3. First we add the fixed obstacle area to the experimental scene in the second experiment in Section 4.2. Fig. 8 shows the allocation details of the CBBA algorithm with obstacle avoidance. We set the obstacle area to a cylindrical shape. In order to visualize the obstacle avoidance result of the algorithm, we only show the path of the agents on the $X$ and $Y$ axes. It can be seen that A2 and A5 have successfully generated the Dubins path to avoid the obstacle area. In order to verify the influence of the obstacle area on the distribution result, we can compare the distribution results of Fig. 8 and Fig. 5. It can be seen that due to the existing obstacle area, the distribution result has changed.

We still tested the total score, calculation time and communication time of the CBBA algorithm

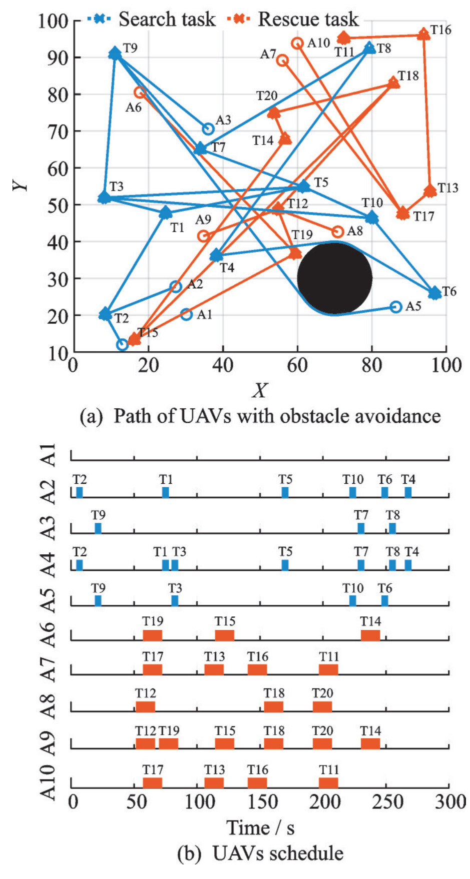
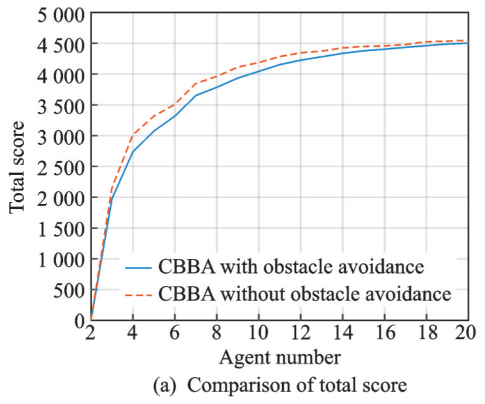


Fig.8 Distribution details of CBBA algorithm with obstacle avoidance (Total ten UAVs, five S-UAVs and five R-UAVs)

with obstacle avoidance. As shown in Fig.9, since the Dubins path to avoid obstacles is calculated, there is a small decrease in the total score, and the calculation time is slightly increased.



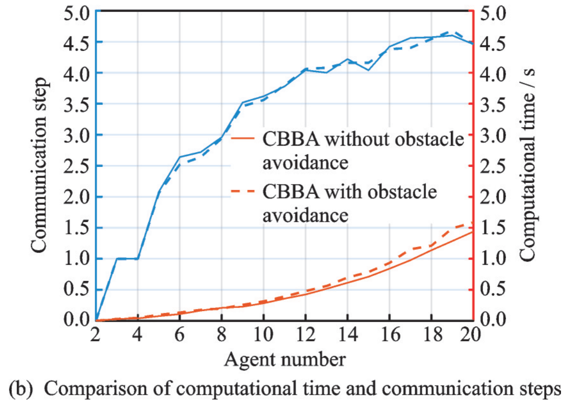


Fig.9 Obstacle avoidance effect on total score, computational time and communication steps

## 5 Conclusions

This paper introduces an extended CBBA algorithm that provides a decentralized task allocation algorithm for heterogeneous UAVs in complex environments. The extension of the algorithm includes enabling the CBBA algorithm to solve multi-agent tasks and obstacle avoidance in complex environments. A new consensus algorithm is proposed to solve the allocation conflict of multi-agent tasks. The winning bid list and the winning agent list in the original CBBA algorithm are combined into a matrix, including all winning information, to solve allocation conflicts caused by multi-agent tasks. Then a more realistic complex environment is considered and the obstacle avoidance is added to the allocation algorithm. Considering the motion constraints of UAV, the Dubins algorithm is used to design a feasible obstacle avoidance path for each UAV.

In this paper, several comparative experiments are carried out to verify the effectiveness of the algorithm. Experiments show that the CBBA algorithm with multi-agent tasks can significantly improve the total task score and does not increase the running time. When obstacles are added to the task scenario, the extended algorithm can design a Dubins path for each UAV to avoid obstacles, although the cost is to reduce the total score and increase the calculation time. In summary, the CBBA extension algorithm proposed in this paper can converge to a conflict-free and feasible solution for decentralized multi-agent task planning problem with obstacle avoidance.

## 6 Further Work

In recent years, we have carried out related technical research and flight tests in the fields of intelligent UAV, autonomous control of UAV swarm and other fields, and have achieved valuable results. Our key research interests include: (1) Low-speed and high-speed UAV platform design, (2) UAV miniaturization design, (3) UAV intelligent perception and cognition, (4) UAV swarm collaborative planning and autonomous decision-making, (5) highly reliable self-organizing network communication for UAV swarm, (6) human-machine collaborative intelligent control, and (7) self-organization control for large-scale UAV swarm.

We carried out the formation experiment of nine fixed-wing UAVs with an air ad hoc network (Fig.10).The tests mainly verified (1) UAV auton-

Fig.10 Formation experiment of nine fixed-wing UAVs with an air ad hoc network

omous air assembly, (2) UAV formation maintenance and transformation, (3) UAV dense formation security control and emergency planning (Fig.11), (4) air ad hoc network communication link, and (5) online task assignment and trajectory planning for UAVs.

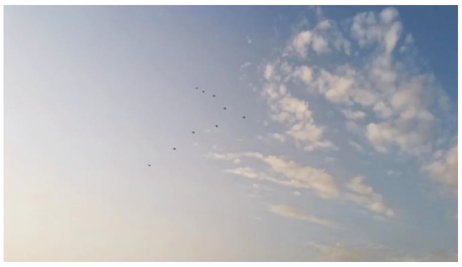


Fig.11 Dense formation security control and emergency planning experiment of nine fixed-wing UAVs

In the near future, a swarm flight test of UAVs will be carried out to perform relevant technical tests in a restricted and jammed environment.

## References

[1] FAHEY K M, MILLER M J. Unmanned systems integrated roadmap 2017—2042[M]. USA: Department of Defense, 2017: 1-5.

[2] WU W, CUI N, SHAN W, et al. Distributed task allocation for multiple heterogeneous UAVs based on consensus algorithm and online cooperative strategy [J]. Aircraft Engineering and Aerospace Technology, 2018, 90(9): 1464-1473.

[3] JOHNSON L B, CHOI H L, HOW J P. The role of information assumptions in decentralized task allocation: A tutorial[J]. IEEE Control Systems, 2016, 36 (4): 45-58.

[4] SCHWARZROCK J, ZACARIAS I, BAZZAN A L C, et al. Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence[J]. Engineering Applications of Artificial Intelligence, 2018, 72: 10-20.

[5] MCLAIN T W, BEARD R W. Coordination variables, coordination functions, and cooperative timing missions[J]. Journal of Guidance, Control, and Dynamics, 2005, 28(1): 150-161.

[6] LIEKNA A, LAVENDELIS E, GRABOVSKIS A. Experimental analysis of contract net protocol in multi-robot task allocation[J]. Applied Computer Systems, 2012, 13(1): 6-14.

[7] KEVICZKY T, BORRELLI F, FREGENE K, et al. Decentralized receding horizon control and coordination of autonomous vehicle formations[J]. IEEE Transactions on Control Systems Technology, 2008, 16(1): 19-33.

[8] LI W, CASSANDRAS C G. Centralized and distributed cooperative receding horizon control of autonomous vehicle missions[J]. Mathematical and Computer Modelling, 2006, 43(9/10): 1208-1228.

[9] DUAN Z, LI W, CAI Z. Distributed auctions for task assignment and scheduling in mobile crowdsensing systems[C]//Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). Atlanta, GA, USA: IEEE, 2017: 635-644.

[10] MEZEI I, LUKIC M, MALBASA V, et al. Auctions and iMesh based task assignment in wireless sensor and actuator networks[J]. Computer Communications, 2013, 36(9): 979-987.

[11] XIAO M, MA K, LIU A, et al. SRA: Secure reverse auction for task assignment in spatial crowdsourcing[J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 32(4): 782-796.

[12] CHOI H L, WHITTEN A K, HOW J P. Decentralized task allocation for heterogeneous teams with cooperation constraints[C]//Proceedings of the 2010 American Control Conference. Baltimore, MD, USA: IEEE, 2010: 3057-3062.

[13] QU G, BROWN D, LI N. Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions[J]. Automatica, 2019, 105: 206-215.

[14] GERKEY B P, MATARIĆ M J. A formal analysis and taxonomy of task allocation in multi-robot systems[J]. The International Journal of Robotics Research, 2004, 23(9): 939-954.

[15] BISWAS S, ANAVATTI S G, GARRATT M A. A time-efficient co-operative path planning model combined with task assignment for multi-agent systems[J]. Robotics, 2019, 8(2): 35.

[16] DUBINS L E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents[J]. American Journal of Mathematics, 1957, 79(3): 497-516.

[17] SHANMUGAVEL M, TSOURDOS A, ZBIKOWSKI R, et al. Path planning of multiple UAVs using dubins sets[C]//Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit. San Francisco, Califomia, USA: AIAA, 2005: 5827.

**Author** Mr. **JIA Tao** received the M.S. degree in

aerospace technology from National University of Defense Technology, Changsha, China. He joined in China Aerodynamics Research and Development Center in February 2014. His research is focused on flight control and UAV swarm.

**Author contributions** Mr. **JIA Tao** designed the algorithm, developed the model architecture, conducted the analysis, interpreted the results and wrote the manuscript. Mr. **XU Haihang** contributed to model components and algorithm simulation. Mr. **YAN Hongtao** contributed to data and flight simulation. Mr. **DU Junjie** contributed to the discussion and background of the study. All authors commented on the manuscript draft and approved the submission.

**Competing interests** The authors declare no competing interests.

(Production Editor: ZHANG Huangqun)

## 异构无人机集群的分布式多智能体任务规划

贾 涛，徐海航，颜鸿涛，杜俊杰
（中国空气动力学研究与发展中心空天技术研究所，绵阳621000，中国）

**摘要：**本文针对具有不同能力的异构无人机集群提出了一种分布式任务规划算法，该算法扩展了基于共识的捆绑算法（Consensus-based bundle algorithm, CBBA），以解决更加现实和复杂的环境。扩展分为两个方面，一方面是处理需要多个无人机协同完成的多代理任务，另一方面是在任务场景中考虑避障路径规划。本文提出了一种新的共识算法来解决多智能体任务分配问题，并使用Dubins算法设计了无人机的避障并考虑运动约束的可行路径。实验结果表明，本文提出的CBBA扩展算法可以得到无冲突、可行的多智能体任务规划解决方案。

**关键词：**任务分配；无人机集群；基于共识的捆绑算法；多智能体任务；避障