The Journal of Supercomputing (2022) 78:10025–10062
https://doi.org/10.1007/s11227-021-04223-3

A resource-constrained distributed task allocation method based on a two-stage coalition formation methodology for multi-UAVs

Mi Yang[1] · An Zhang[1] · Wenhao Bi[1] · Yunong Wang[1]

Accepted: 17 November 2021 / Published online: 20 January 2022

Abstract

The task allocation problem is an important research field in unmanned aerial vehicles (UAVs). However, most existing task allocation algorithms can form coalitions to address the resources constraints, but cannot support starting tasks at the same time, nor can cope with the new emerging tasks flexibly. To this end, we propose a novel resource-constrained task allocation method based on the performance impact algorithm (RCPIA) to support simultaneously starting tasks and provide more flexibility to reallocate the new tasks. More specifically, based on the proposed task allocation model, we firstly modify the task inclusion phase and conflict resolution phase of the baseline PI algorithm to preferentially allocate the tasks to the UAVs that can complete tasks individually. After that, to make full use of resources and further allocate remaining unassigned tasks, a two-stage coalition formation method is creatively proposed to form a coalition for the tasks that cannot be performed by a single UAV to provide enough resources. Especially, an idle time slot mechanism (ITSM) is investigated to shift the start times of tasks that can be performed by a single UAV to create a longer feasible time slot to insert the task. Thirdly, the reassignment application of the two-stage coalition method is introduced to cope with new emerging tasks. Finally, numerical simulations are constructed to illustrate the procedure of RCPIA and verify the superiority of RCPIA compared with other task allocation algorithms in efficiency and success allocation rate.

Keywords Multi-UAV System · Distributed task allocation · Resource constraint · Coalition formation · Dynamic environment

This work was supported by the National Natural Science Foundation of China (No. 61903305, No. 62073267), the Aeronautical Science Foundation of China (No. 201905053001), and the Research Funds for Interdisciplinary Subject, NWPU.

✉ An Zhang
zhangan@nwpu.edu.cn

[1] School of Aeronautics, Northwestern Polytechnical University, Xi'an 710072, Shaanxi, China

1 Introduction

A multiple unmanned aerial vehicles (multi-UAV) system is a set of UAVs that are designed to cooperate with each other to complete some complex missions [22, 30, 32]. The advantages of multi-UAV system lie in increased flexibility, enhanced reliability and resilience, simultaneous broad area coverage or capability to operate outside the communication range of base stations [5, 24, 40]. Hence, multi-UAV systems have been applied to many real-world scenarios [21, 43].

Task allocation means assigning a set of tasks to UAVs without conflicts and constraints violation while optimizing some objective function [36, 45]. Hence, the solution that does not violate constraints is vital to improve the performance of completing complex missions [11]. Much research has studied dealing with complex constraints in problems, such as the constraints of limited resource, resource type, time windows and reassignment version in dynamic environment [20, 38].

However, there are many challenges that exist in addressing the complex constraints of multi-UAV systems task allocation [36, 45]. In real-world, the resources carried by heterogeneous UAVs are diverse and limited, while the tasks also may require different types of resources. In addition to the resource type constraints, we hope to make full use of the limited heterogeneous resources of UAVs to further allocate more tasks. Hence, it is allowed to select a group of UAVs to cooperatively provide enough resources for executing one task. Besides, due to the military requirement, some tasks must be started at the same time to guarantee the task completion performance. Since starting tasks at the same time will affect the start time of other tasks in the task list, which may cause the objection function of the solution to decrease. Therefore, it is necessary to develop approaches that form a group of UAVs to provide enough resources for tasks and cooperatively start a task at the same time without much objective function loss. Moreover, the new emerging tasks may have a greater impact on the coalition UAVs than single UAV. Once the task list of a coalition UAV is interrupted by the new task, other coalition member UAVs must wait for its arrival. This will cause the allocated tasks cannot be performed on time or not be executed. Hence, how to provide a feasible and flexible solution in a dynamic environment within a short time is much worth studying.

To this end, we focus on the distributed task allocation problem where heterogeneous UAVs are allocated to several tasks with resource-type constraints and the simultaneous starting tasks should also be supported. In this paper, it is preferred to avoid unnecessary cooperation of UAVs and alleviate the impact of new emerging tasks on preallocation solutions. Also, to provide greater flexibility, more redundant time should be fully utilized to create feasible time slots by shifting the start times of a single UAV executing tasks. Therefore, we propose a novel resource-constrained task allocation algorithm based on the performance impact algorithm (RCPIA). In detail, this paper makes the following contributions to the state of the art:

- The task assignment problem with resources type constraints and limited quantity constraints is modeled in the mathematical formulation. Besides, the requirement of starting coalition tasks concurrently is also taken into consideration.

- To avoid unnecessary cooperation, it is first preferred to allocate a single UAV with sufficient resources to the tasks, rather than form a coalition to complete tasks to obtain a higher reward. To this end, this paper first modifies the criteria of adding tasks and removing tasks to allocate the tasks to the UAVs with enough resources which can complete them individually as much as possible. Then, the conflict resolution rules are also modified to meet the resource constraints.
- To further make full use of UAV resources, a two-stage coalition formation method is introduced to further allocate the tasks that cannot be performed by a single UAV. First, in the first stage, the UAVs that can provide the required resources for the task and meet the deadline constraints are selected as preliminary coalition members. It's worth noting that we propose an idle time slot mechanism (ITSM) to compute the feasible time slot for the UAV to insert the task. This mechanism can calculate the start time redundancy for the current coalition task without affecting the start times of other coalition tasks in the task list. After selecting preliminary coalition members, the second stage is used to further reduce the coalition size based on three different criteria and the final coalition members are obtained.
- In order to cope with the new emerging tasks in a dynamic environment, the reassignment application of the two-stage coalition formation method is investigated.

The rest of the paper is organized as follows. In Sect. 2, we overview the previous works related to the task allocation problem of multi-UAV systems. Section 3 introduces the resource-based multi-UAV task allocation problem and gives its mathematical formulation. The main structure of RCAPI is illustrated in Sect. 4. Several simulations and comparisons are conducted in Sect. 5 to demonstrate the performance of the RCAPI. Section 6 concludes this paper.

2 Related work

There have been numerous studies on the task allocation problem of multi-UAV systems and many methods are proposed for the past few years. Generally, they are categorized into two main categories: optimized-based and market-based [3].

Many optimized-based methods are proposed to find the optimal solution. For example, ant colony algorithm (ACO) [34, 46] and practical swarm optimization (PSO) [25, 28] were developed due to their fast convergence speed in solving large-scale task allocation problems. Besides, as a typical optimization-based approach, genetic algorithm (GA) is also one of the most useful methods to solve task assignment problem when the search space is not extremely rugged in the presence of many types of constraints [45]. Many algorithms are incorporated into GA to improve the efficiency or avoid the local optimum. In [35], the improved simulated annealing algorithm (SA) was used in the second selection operation to improve the population diversity of GA. [37] proposed a multi-objective shuffled frog-Leaping algorithm (MOSFLA) and GA-based task assignment and sequencing method which

showed that the total operation time shrinks. However, since optimized-based algorithms are difficult to design appropriate local decision rules, they are usually used in centralized systems. This places a heavy communication burden on the server and also reduces the mission range. In addition, they are vulnerable to the single point of failure and are easy to fall into local optimum.

Hence, as a typical distributed approach, the market-based algorithm [29, 31, 41, 44] has been applied to achieve a suboptimal solution due to its low computation complexity, efficiency, robustness, and scalability. Recently, the consensus-based bundle algorithm (CBBA) has attracted considerable research interest since it combines the positive properties of the auction and conflict resolution to produce a conflict-free assignment [8]. It has been proven that this method can offer similar solutions to some centralized sequential greedy algorithms and 50% optimality is guaranteed. Several researches made modifications and extensions based on CBBA [27, 38], and several algorithms were also inspired from it [13, 33, 46] which also performed well in solving task allocation problems. Besides, many other algorithms were also applied to solve such problems, see [1, 2, 12, 18, 19, 26].

For dealing with resource constraints, [17] adopted cross-entropy (CE) to address the resource required by tasks and demonstrated its effectiveness. A greedy heuristic was introduced in [42], which considered inter-task resource constraints to approximate the influence between different assignments in task allocation. [23] proposed a distributed task allocation method based on resource welfare by balancing resource depletions. [10] designed a modified genetic algorithm with multi-type genes for the task assignment with limited resources and kinematic constraints. However, the aforementioned algorithms just considered assigning one task to one UAV, which can not make full use of the resources of heterogeneous UAVs.

At the same time, most existing works studied forming coalition UAVs to cooperatively complete tasks, which aims to obtain a better solution compared to assigning one UAV to one task [9, 14, 39], and only a few studies considered both resource constraint and coalition formation. For example, [7] introduced a holistic coalition methods for global optimization and a sequential coalition method to select suitable robots to form coalitions for the dynamic events. Similarly, a leader-follower coalition methodology (LFCM) for solving resource constraints was discussed in [6]. Nevertheless, these two methods were both applied in a centralized way and did not consider the constraints of simultaneously starting tasks. In addition, they cannot allocate the new tasks in real-time. [4] proposed a coalition formation method based on game theory (CFGT) to provide enough resources for tasks. Similarly, this method also cannot start tasks at the same time. [13] proposed a resource dynamic assignment algorithm inspired by CBBA based on task sequence mechanism (RDAATSM) and the simulation verified that the algorithm can assign new tasks with limited resources online. The task sequence mechanism can insert the new tasks into its task list without affecting the allocated tasks. To perform tasks in the shortest time and occupy fewer UAVs resources, RDAATSM forms coalitions for each task and there is synchronous waiting time generated to ensure the task can be started at the same time. However, this may lead to three troubles. The first is that only inserting the new tasks into the idle period (synchronous waiting time) will reduce the flexibility and success rate to allocate new tasks since the preallocated start times are not

allowed to change. The second is that if the assigned task encounters dynamic events (such as the task execution time extends, the last task is delayed, the task disappears, etc.), these dynamic events have a greater impact on the task execution of subsequent tasks for all coalition UAVs than on the UAV executing the task independently. As for the third weakness, it is known that the CBBA algorithm must meet the marginal diminishing characteristics of the task reward during the bundle construction phase. Therefore, RDAATSM is designed to assign the most resources to the tasks with the greatest reward, which means that this algorithm first selects tasks to insert according to the greatest reward then according to the provided resources. This will lead to more tasks being performed by a coalition rather than a single UAV. In brief, we provide a comparative analysis of five typical task allocation algorithms, which are PI ([43]), cross-entropy (CE) [17], Leader-Follower Coalition Methodology (LFCM) ([7]), coalition formation method based on game theory (CFGT)([4]) and Resource Dynamic Assignment Algorithm Based on Task Sequence Mechanism (RDAATSM) ([13]). Comparison are analyzed from: (i) resource constraints; (ii) dynamic assignment; (iii) implementation way; (iv) simultaneous start; and (v) objective. The comparative analysis is summarized in Table 1.

Hence, this paper investigates a distributed algorithm to solve the task allocation problem with constraints of resource and simultaneous start. In addition, this algorithm should be flexible to address the new emerging tasks in a dynamic environment by slightly changing the original task allocation scheme.

3 Problem description

Consider a scenario where N_u heterogeneous UAVs $\mathbf{V} = [v_1, \ldots, v_{N_u}]^T$ are allocated to execute N_t heterogeneous tasks $\mathbf{T} = [t_1, \ldots, t_{N_t}]^T$, with $N_t > N_u$. Heterogeneous tasks are distinguished by the number of different resources required by the task. A task allocation solution $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{N_u}]^T$ consists of the assignments for all UAVs, where $\mathbf{a}_i = [t_{i1}, \ldots, t_{i|\mathbf{a}_i|}], i = 1, \ldots, N_u$ is the task list assigned to UAV v_i and is ordered by the actual start time of assigned tasks. The size of a task list $\mathbf{a}_i$ is decided by the number of tasks assigned to v_i. In addition, each task has a latest start time (deadline) $\mathbf{S} = [s_1, \ldots, s_{N_t}]^T$. $\mathbf{G}(t)$ denotes a symmetric communication matrix, where $g_{i,j}(t) = 1$ indicates that UAV v_i can directly communicate with UAV v_j at time

Table 1 Comparative characteristics of previous works

1	PI [43]	CE [17]	LFCM [7]	CFGT [4]	RDAATSM [13]
Resource constraints	–	√	√	√	√
Dynamic assignment	–	–	–	√	√
Implementation way	Distributed	Centralized	Centralized	Hybrid	Distributed
Simultaneous start	–	–	–	–	√
Objective	Minimizing waiting time	Maximizing global reward	Maximizing global reward	Maximizing global reward	Maximizing global reward

t and we denotes the UAVs as neighbors. A list of key symbols used is provided in Table 2.

3.1 Resource modeling

(1) The UAV is a physical entity with given resource capabilities that can be used to process tasks. A resource is a measurable physical or virtual entity used in the processing of tasks. Denote the resources vector R_i^U on UAV v_i as follows:

Table 2 Symbol Definitions

Symbol	Definition
N_u	Number of heterogeneous UAVs
N_t	Number of tasks
$\mathbf{a}_i$	Ordered task list of UAV v_i
C_i	Final coalition for task t_i
$\tau_{ij}^{\star}$	Time for UAV v_i to reach task t_j
τ_{ij}	Actual start time for UAV v_i performing task t_j
τ_j	Actual start time for task t_j
s_k	Deadline (latest start time) for t_k
$\mathbf{G}$	Symmetric communication matrix
$R_i^U = [R_{i1}^U, \ldots, R_{iM}^U]$	Resources carried by UAV v_i
$R_i^T = [R_{i1}^T, \ldots, R_{iM}^T]$	Resources required to complete the task t_i
SV_k	Static value of task t_k
η_{ij}	The ratio of the number of resources v_i can provide to the number of resources required by t_j
d_k^E	Execution time of t_k
$d_{i,k,k+1}^T$	Travel time of v_i form t_k to t_{K+1}
$d_{i,k}^W$	Waiting time for v_i to start t_k
ψ	The moment when a dynamic event occurs
$\mathbf{a}_i \ominus t_k$	$\mathbf{a}_i$ with t_k removed
$\mathbf{a}_i \oplus_l t_k$	The inclusion of t_k at position l in $\mathbf{a}_i$
$\mathbf{a}_i^{\ominus k}$	Temporary task list with t_k removed
$w_{i,k}^{\oplus}$	The inclusion performance impact (IPI) of t_k in $\mathbf{a}_i$
$w_{i,k}^{\ominus}$	The removal performance impact (RPI) of t_k in $\mathbf{a}_i$
$w_{i,u}^{\ominus\diamond}$	The global winning RPI value to record the highest RPI value to remove t_u in coalition C_u
$\beta_{i,u}^{\diamond}$	The updated winning UAV ID to record the winning UAV of task t_u
$\boldsymbol{\gamma}_i^{\ominus} = [w_{i,1}^{\ominus}, \ldots, w_{i,N_t}^{\ominus}]$	A global winning RPI list for all tasks recorded by UAV v_i
$\boldsymbol{\gamma}_i^{\ominus\diamond} = [w_{i,1}^{\ominus\diamond}, \ldots, w_{i,N_t}^{\ominus\diamond}]$	An updated global winning RPI list for all tasks
$\boldsymbol{\beta}_i = [\beta_{i,1}, \ldots, \beta_{i,N_t}]$	A winning UAV ID list corresponding to the RPI list that records which task is assigned to which UAV in v_i's local view
$\boldsymbol{\beta}_i^{\diamond} = [\beta_{i,1}^{\diamond}, \ldots, \beta_{i,N_t}^{\diamond}]$	An updated global winning UAV ID list records which task is assigned to which UAV

$$R_i^U = [R_{i1}^U, \ldots, R_{iM}^U] \tag{1}$$

where $R_{ip}^U, p = 1, \ldots, M$ denotes the amount of the pth type resource carried by UAV v_i and M is the number of supplied resource types. For example, $R_i^U = [2, 0, 1]$ means that UAV v_i carries two Resource 1 and one Resource 3, and does not carry Resource 2 or Resource 2 has been used up. It is noted that the resources carried by UAVs will constantly be consumed as the tasks are processed.

(2) The tasks are derived by decomposing a high-level task and are activities that require relevant resources that will be provided via UAVs. Denotes R_i^T as the required resources for completing task t_i as follows:

$$R_i^T = [R_{i1}^T, \ldots, R_{iN}^T] \tag{2}$$

where $R_{iq}^T, q = 1, \ldots, N$ denotes the number of the qth type of resources required by task t_i, and N represents the number of types required by the task. For simplicity, $M = N$ is adopted in this paper.

In order to complete the task t_i, it is required that the assigned coalition (maybe a single UAV) must provide all types and enough amounts of resources required by the task. The relationship between the resources, tasks and UAVs is shown in Fig. 1.

3.2 Utility function

In order to achieve a feasible solution, the overall optimization objective is defined as follows:

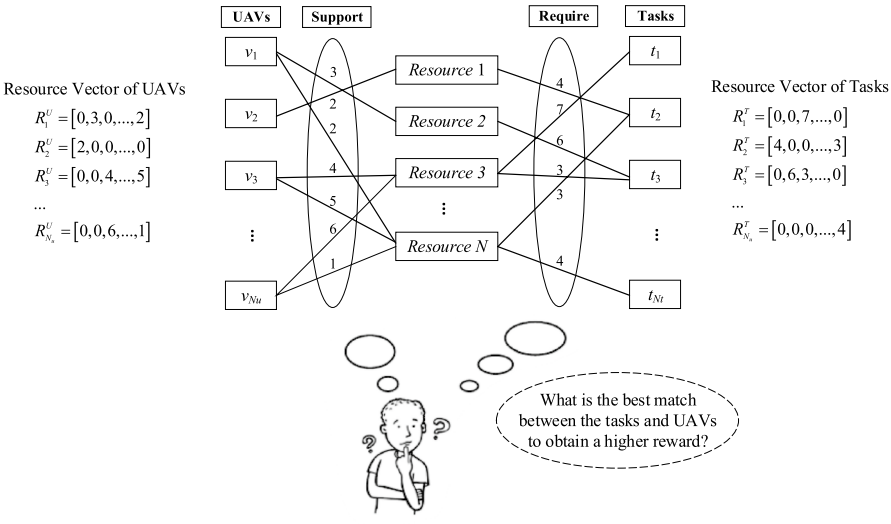

Fig. 1 The relationship between the resources, tasks and UAVs

$$max\{J = \sum_{i=1}^{N_u} \sum_{j=1}^{N_T} X_{i,j} \frac{s_j - \tau_{ij}}{s_j} SV_j \eta_{ij}\} \tag{3}$$

subject to:

$$\begin{aligned} &\sum_{i=1}^{N_u} X_{i,j} R_{ik}^U \geq R_{jk}^T, \forall j \in T, \forall k \in N \\ &X_{i,j} \tau_{ij} \leq s_j, \forall j \in T \\ &\tau_j = X_{i,j} \tau_{ij}, \forall j \in T \\ &X_{i,j} \in \{0,1\}, \forall (i,j) \in V \times T \end{aligned} \tag{4}$$

where$X_{i,j}$ denotes whether UAV v_i is assigned to task t_j. The static reward SV_j is defined as the contribution of completely completing the task to the overall goal, and we can also regard the static reward as the inherent importance of the task t_j. A linear time-discount strategies is used here to indicate the effect of start time on task reward. η_{ij} is defined as the contribution proportion, i.e., the ration of the number of resources that UAV v_i can provide to the number of resources required by the task t_j. The constraints described in Eq. (4) means as follows: the resources that UAVs provide must cover the task required resources, the actual start time of a task must not be later than its deadline, the coalition for a task must start the task at a same time.

The actual start time τ_{ij} is computed as Eq. (5) based on the order of task execution in task list $\mathbf{a}_i$,

$$\tau_{ij} = \tau_j = \max_{i=1}^{|C_j|} \tau_{ij}^*, v_i \in C_j \tag{5a}$$

$$\tau_{ij}^* = \begin{cases} d_{i,o,j}^T, & j = 1 \\ \tau_{i(j-1)} + d_{j-1}^E + d_{i,j-1,j}^T, & 1 < j \leq |\mathbf{a}_i| \end{cases} \tag{5b}$$

$$\tau_{ij} = \tau_{ij}^* + d_{i,j}^W \tag{5c}$$

where $d_{i,o,j}^T$ denotes the travel time from the initial location to the first task in $\mathbf{a}_i$ for v_i and $d_{i,j}^W$ denotes the waiting time of UAV v_i to task t_j. d_j^E represents the execution time for performing t_j. τ_j represents the actual start time for task t_j and $\tau_{i,j}^*$ denotes the actual arrival time for v_i to t_j.

If task t_j is assigned to a coalition C_j, in order to guarantee the performance of task completion, the UAV needs to wait for all other members of the coalition to arrive before they can start to execute the task. Hence, Eq. (5a) shows that the time cost of v_i starting t_j is equal to the maximum arrival times for all UAVs in coalition C_j and this maximum arrival time is also regarded as the real start time for the task t_j. Eq. (5b) means that the arrival time for UAV v_i to reach task t_j is computed as the sum of the travel times and execution times of all previous tasks. It is noted that time costs are cumulative so that the cost of servicing task t_j includes the time cost of

servicing all the tasks previous to it in the task list. Eq. (5c) demonstrates the relationship between the actual start time and arrival time for v_i to t_j. It is noted that, if task t_j is assigned to a single UAV v_i, the UAV starts t_j as soon as it arrives at t_j, i.e., $\tau_j = \tau_{ij} = \tau^*_{ij}$. The schematic representation of time cost is expressed in Fig. 2, where v_k is the last UAV to reach t_j.

The contribution proportion η_{ij} is defined as the ration of the number of resources that UAV v_i can provide to the number of resources required by the task t_j as follows:

$$\eta_{ij} = \frac{\sum_{k=1}^{M} min(R^U_{ik}, R^T_{ik})}{\sum_{k=1}^{M} R^T_{ik}} \tag{6}$$

For example, when task t_j requires the resources $R^T_j = [0, 3, 1]$ and UAV v_i carries the resources $R^U_i = [1, 3, 2]$, t_j can be completed by v_i individually, i.e., $\eta_{ij} = 1$. If $R^U_i = [0, 2, 0]$, then v_i can only contribute two resources of four resources required by t_j and η_{ij}=0.5.

4 Proposed method

In order to address the resource constraints on heterogeneous UAVs in a distributed way, we propose a resource-constrained task allocation method based on the performance impact algorithm (RCPIA) by modifying the criteria to include and remove tasks in [43] and introducing a two-stage coalition formation method. The main program of RCPIA is described as Algorithm 1.

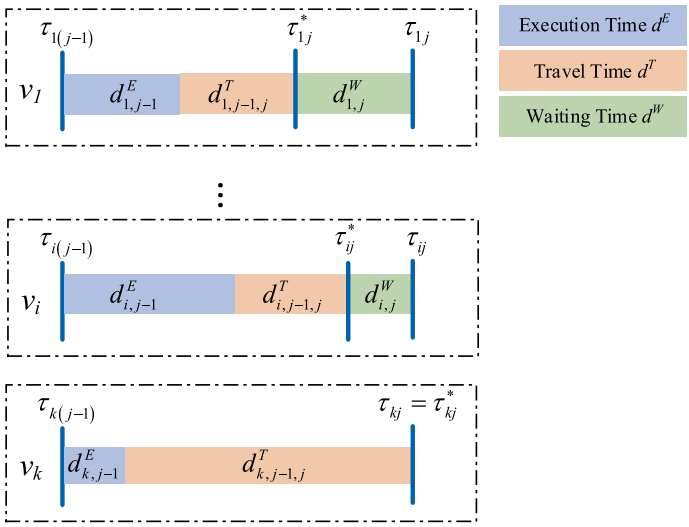

Fig. 2 The relationship between the execution time, travel time and waiting time

In more detail, after initializing the mission environment, all UAVs first communicate with neighbors to achieve consensus regarding the winning UAVs and winning RPIs for all tasks (line 4). Then, for each UAV, the conflict resolution phase is first carried to remove the conflict tasks (line 6) and the task inclusion phase is followed to further include more tasks into the task list until no more tasks can be included (line 7). After all, UAVs have completed a cycle of these two phases, if no more tasks can be swapped among all UAVs to maximize the total score, the algorithm ends. Otherwise, a new communication is triggered again. So far, a conflict-free and feasible solution has been obtained. Some tasks are assigned to the UAVs that can complete them individually. In addition, the leaders which can provide the most resources for the tasks that cannot be performed by a single UAV are also obtained. Then, the two-stage coalition formation method is followed to form a coalition for the tasks that cannot be completed by a single UAV (line 12).

Algorithm 1 RCPIA Main Program

```
1: Define World, UAVs, Tasks
2: Initialize Timer T ← 1
3: while converged is false do
4:    Communication
5:    for each UAV do
6:       Conflict resolution phase
7:       Task inclusion phase
8:    end for
9:    converged ← Check convergence
10:   T ← T + 1
11: end while
12: Two-stage coalition formation method
```

4.1 Main structure

4.1.1 Task inclusion phase

During the task inclusion phase, every UAV tries to include tasks into its task list until no more tasks can be included or no more resources are left based on the knowledge of the resource requirements for all tasks. At the beginning of RCPIA, the task list $\mathbf{a}_i$, local RPI list W_i^{Θ}, global winning UAV list $\boldsymbol{\beta}_i$ and global winning contribution proportion list $\mathfrak{V}_i^{\diamond}$ for all UAVs are initialized as empty and set as the initial inputs of RCPIA. Once the conflict resolution has been carried, the inputs change to the conflict-free outputs of the resolution phase. It is noted that, since this phase is locally carried by each UAV, the output solution is maybe conflicting. The pseudo-code of this phase is shown in Algorithm 2.

Algorithm 2 Task Inclusion Phase for UAV v_i

Input: Tasks $\mathbf{T}$, UAVs $\mathbf{U}$, tasks lists $\mathbf{a}_i$, local RPI list $W_i^{\ominus}$, winning UAV list $\boldsymbol{\beta}_i$, local contribution proportion list $\mho_i$, global winning contribution proportion list $\mho_i^{\diamond}$ and current resources R_i^U for all UAVs

Output: Task lists $\mathbf{a}_i$, global winning RPI list $W_i^{\ominus\diamond}$, winning UAV list $\boldsymbol{\beta}_i$, global winning contribution proportion list $\mho_i^{\diamond}$ and remaining resources R_i^U for all UAVs

1: **while** $|\mathbf{a}_i| \leq N_t$ **do**
2: **for** $t_k \in \mathbf{T}$ **do**
3: **if** $t_k \notin \mathbf{a}_i$ **then**
4: Compute η_{ik} as Eq. (6) // *Compute the contribution degree of* v_i *to* t_k
5: Compute $w_{i,k}^{\oplus}$ from Eq. (7) and (8) // *Compute the IPI of* v_i *including* t_k
6: **end if**
7: **end for**
8: $\mho_i = [\eta_{i1}, ..., \eta_{iN_t}]$ // *Form the global contribution proportion list*
9: $T_i^A = \{t_g | \arg\max_{k=1}^{N_t} \mho_i, \eta_{ig} > 0$, and $w_{i,k}^{\oplus} > 0\}$ // *Form the candidate tasks set for* v_i *to add*
10: **if** $|T_i^A| = 1$ **then**
11: $\mathbf{a}_i \leftarrow \mathbf{a}_i \oplus_l t_g$ // *Insert the unique task* t_g *with maximum contribution degree into its task list*
12: **else if** $|T_i^A| > 1$ **then**
13: Compute the local RPI list $W_i^{\ominus}$ for all tasks by Eq. (10)
14: $t_g \leftarrow \arg\max_{k=1}^{T_i^A} \left\{ w_{i,k}^{\oplus} - w_{i,k}^{\ominus} \right\}$ by Eq. (11) // *Select the task* t_g *with maximum difference between IPI and RPI from the candidate tasks se* T_i^A
15: $\mathbf{a}_i \leftarrow \mathbf{a}_i \oplus_l t_g$ //*Insert* t_g *into its task list*
16: **end if**
17: $w_{i,g}^{\ominus} \leftarrow w_{i,g}^{\oplus}$ // *Set the RPI of* t_g *as the IPI of* v_i *including* t_g
18: Update $W_i^{\ominus}$, $W_i^{\ominus\diamond}$, $\boldsymbol{\beta}_i$, $\mho_i^{\diamond}$ and R_i^U
19: **end while**

- Step 1(lines 2–7) At the beginning, the contribution proportion η_{ij} for each task is also computed by Eq. (6).
  Then, the inclusion performance impact (IPI) is defined to measure the newly added task's local performance impact generated by the UAV. It is defined as the greatest difference in the score (with and without the added task) of the added task and subsequent tasks for all possible insert positions. Every UAV computes the IPIs for all tasks as shown in Eqs. (7) and (8) (line 5):

$$w_{i,k,l}^{\triangle} = u_{i,k}\left(\mathbf{a}_i \oplus_l t_k\right) + \sum_{r=l}^{|\mathbf{a}_i|} \left(u_{i,r+1}\left(\mathbf{a}_i \oplus_l t_k\right) - u_{i,r}\left(\mathbf{a}_i\right)\right) \tag{7}$$

$$w_{i,k}^{\oplus} = \max_{l=1}^{|\mathbf{a}_i|+1} \left\{ w_{i,k,l}^{\triangle} \right\} \tag{8}$$

where $\mathbf{a}_i \oplus_l t_k$ denotes adding the task t_k into the task list $\mathbf{a}_i$ of UAV v_i at l position. When the task has been included in $\mathbf{a}_i$ or the UAV cannot provide enough resources required by the task, the IPI is set to 0. A local IPI list $W_i^{\oplus} = [w_{i,1}^{\oplus}, \ldots, w_{i,N_t}^{\oplus}]$, $i = 1, \ldots, N_u$ is designed to store the IPIs of UAV v_i including all tasks.
  It is known that completing a new task which inserted in the UAV's task list can increase the local score of the UAV. However, the time costs of subsequent tasks

in the task list may be delayed since subsequent tasks need to be shifted to create enough time to execute the new task. From Eq. (2), the delay time costs of subsequent tasks may lead to the decrease of the local scores of the UAV performing subsequent tasks. Hence, as shown in Eq. (7), the local performance impact $w^{\Delta}_{i,k,l}$ of inserting t_k into position l of $\mathbf{a}_i$ is defined as the local score of v_i executing t_k plus the sum of decrease in local scores of other tasks in $\mathbf{a}_i$ that have been assigned previously. Eq. (8) selects the maximum local performance impact for all possible positions as the IPI $w^{\oplus}_{i,k}$ of task t_k in $\mathbf{a}_i$.

- Step 2 (lines 8–9) The contribution proportions for all tasks are stored in a local contribution proportion list $\mathfrak{V}_i = [\eta_{i1}, \ldots, \eta_{iN_t}]$. In order to avoid unnecessary coordination and communication, it is preferred to allocate the task to a single UAV rather than a coalition. Hence, the tasks with the highest contribution proportion and positive IPI are selected as the candidate tasks set to add, i.e., T^A_i, as Eq. (9):

$$T^A_i = \{t_k | \arg\max_{k=1}^{N_t} \eta_{ik}\} \tag{9}$$

- Step 3 (lines 10–16) If $|T^A_i| = 1$, the task t_g with the highest contribution degree is selected to be inserted into the task list $\mathbf{a}_i$ (line 11) and goes to Step 4.

However, if there are more than one tasks in T^A_i, i.e., $|T^A_i| > 1$, the task with the highest improvement for the global objective is selected to add to the task list. However, if the task has been previously assigned to a UAV, the performance impact of adding it should not only consider the IPI but also consider the local impact of removing the task from the previously assigned task list.

The RPI is introduced to measure the performance impact of removing a task and is defined as the greatest difference in the score (with and without the added task) of the deleted task and subsequent tasks as Eq. (10):

$$w^{\ominus}_{i,k} = u_{i,b}(\mathbf{a}_i) + \sum_{r=b+1}^{|\mathbf{a}_i|} \left(u_{i,r}(\mathbf{a}_i) - u_{i,r}(\mathbf{a}_i \ominus t_k)\right) \tag{10}$$

where $\mathbf{a}_i \ominus t_k$ denotes the removal of task t_k from the task list $\mathbf{a}_i$ of UAV v_i, and b symbolizes the position of task t_k in the task list, i.e., $a_{i,b} = t_k$.

Although removing a task from the task list will cause the loss of the local score of UAV performing the removed task, the start times of subsequent tasks will be advanced due to removing the task. The local scores of subsequent tasks will be increased from Eq. (3). Hence, as shown in Eq. (10), the RPI of removing task t_k from $\mathbf{a}_i$ is computed as the loss of executing the removed task t_k plus the increased local scores of subsequent tasks in task list $\mathbf{a}_i$. It represents the contribution of a task to the local score generated by a UAV.

The RPIs of all unassigned tasks are initialized as 0. Hence, the RPI must be greater than 0 once be assigned. The RPIs of all tasks for UAV v_i are stored in a local RPI list $W^{\ominus}_i = [w^{\ominus}_{i,1}, \ldots, w^{\ominus}_{i,N_t}], i = 1, \ldots, N_u$. To facilitate consensus, defines a local winning UAV list $\boldsymbol{\beta}_i = [\beta_{i,1}, \ldots, \beta_{i,N_u}]^T$ to represent which UAV is assigned to which task for UAV v_i's the local view, and $\beta_{i,k}$ is set as the ID of the UAV assigned to task t_k.

Based on the definitions of RPI and IPI, it can be inferred that the global score can be improved by reallocating t_k to UAV v_i if the IPI of task t_k in $\mathbf{a}_i$ is higher than t_q's RPI in another UAV's task list $\mathbf{a}_j$. Therefore, decide which task can be inserted into the current the current task list $\mathbf{a}_i$ of v_i according to Eq. (11).

$$g = \max_{k=1}^{T_i^A} \left\{ w_{i,k}^{\oplus} - w_{i,k}^{\ominus} \right\} \tag{11}$$

If $g > 0$, the task $t_g = \arg\max_{k=1}^{T_i^A} \left\{ w_{i,k}^{\oplus} - w_{i,k}^{\ominus} \right\}$ with the greatest difference of IPI and RPI in current candidate tasks set, i.e., the task can provide the greatest objective improvement, should be selected to insert into $\mathbf{a}_i$ at the position $l_{t_g}^i = \arg w_{i,k}^{\oplus}$. If $g \leq 0$, IPIs of all tasks are equal or lower to the current RPIs, which means that the current assignment cannot be improved or the constraints cannot be met. In this case the task inclusion phase ends.

Step 4 (lines 17–18) The final RPI is set equal to the IPI. In addition, the local RPI list $W_i^{\oplus}$, the global winning RPI list $W_i^{\oplus\diamond}$, winning UAV list $\boldsymbol{\beta}_i$, remaining resources of UAV R_i^U and global winning contribution proportion list $\mathfrak{O}_i^{\diamond}$ are updated. Repeat this inclusion process until no more tasks can be included.

4.1.2 Communication

Because every UAV locally builds its own task list, two or more UAVs may have a conflict with the winning UAV for each task. To this end, all UAVs must communicate with other UAVs to achieve the consensus on the global winning UAVs, global winning RPIs, and global winning contribution proportions for all tasks.

First, all UAVs communicate with each other where a communication link exists between them based on a network topology to transmit some fundamental information, i.e., the local RPI list $W_i^{\ominus}$, winning UAV list $\boldsymbol{\beta}_i$, the local contribution proportion list $\mathfrak{O}_i$ and time stamps TS_i that stores the iteration numbers of the last information update from each of the other UAVs.

Then, after receiving the information from other UAVs, the receiver UAV uses the consensus procedure to achieve global consensus in terms of the aforementioned four lists. In detail, the receiver can determine three types of operation shown in Eq. (12) (update, leave and reset) by evaluating four lists with a decision rule table referred to the Appendix Table 11. For example, when v_i receives information about task t_k from sender v_j, three possible actions are displayed as follows:

$$\begin{aligned}
&\text{Update}: \beta_{i,u} = \beta_{j,u}, w_{i,u}^{\ominus\diamond} = w_{j,u}^{\ominus\diamond}, \eta_{i,u}^{\diamond} = \eta_{j,u}^{\diamond} \\
&\text{Reset}: \beta_{i,u} = 0, w_{i,u}^{\ominus\diamond} = \phi, \eta_{i,u}^{\diamond} = 0 \\
&\text{Leave}: \beta_{i,u} = \beta_{i,u}, w_{i,u}^{\ominus\diamond} = w_{i,u}^{\ominus\diamond}, \eta_{i,u}^{\diamond} = \eta_{i,u}^{\diamond}
\end{aligned} \tag{12}$$

Since the tasks are preferred to allocate to the UAV with a higher contribution proportion, the contribution proportion is firstly compared and the UAV with the higher contribution proportion is selected as the winner UAV. When multiple UAVs can provide the same contribution proportion, the RPI value is utilized to further select

the winning UAV, which means that the UAV with a higher RPI becomes the winner UAV.

After updating the winner UAV, the global winning contribution proportion and RPI for the task are also updated. It is noted that the consensus of the global winning RPI list and global winning contribution proportion list are additionally stored in $W_i^{\ominus\diamond}$ and $\mho_i^{\diamond}$ respectively, while $\boldsymbol{\beta}_i$ held by v_i is directly updated. The local winning RPI list $W_i^{\ominus}$ and local winning contribution proportion list $\mho_i$ remain unchanged in each communication process.

4.1.3 Conflict resolution phase

Algorithm 3 Conflict Resolution Phase for UAV v_i

Input: Task lists $\mathbf{a}_i$, local RPI list $W_i^{\ominus}$, global winning RPI list $W_i^{\ominus\diamond}$, global winning UAV list $\boldsymbol{\beta}_i$, local contribution proportion list $\mho_i$, global winning contribution proportion list $\mho_i^{\diamond}$ and remaining resources R_i^U for all UAVs

Output: Task lists $\mathbf{a}_i$, local RPI list $W_i^{\ominus}$, global winning RPI list $W_i^{\ominus\diamond}$, global winning UAV list $\boldsymbol{\beta}_i$, global winning contribution proportion list $\mho_i^{\diamond}$ and remaining resources R_i^U for all UAVs, actual arrival times for all tasks $\boldsymbol{\tau}^{\star}$

$T_i^R = \{\mathbf{a}_i | \boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i\}$ // *Find candidate removal tasks set*
while $T_i^R \neq \phi$ **do**
  Compute h for tasks in T_i^R from Eq. (14) //*Calculate the maximum difference of the local contribution degree and global contribution degree*
  if $h < 0$ **then**
    Break
  end if
  $T_i^{R'} = [t_k | \arg h, t_k \in T_i^R]$ // *The tasks with same maximum difference of the local contribution degree and global winning contribution degree form the candidate set* T_i^R
  if $|T_i^{R'}| = 1$ **then**
    $\mathbf{a}_i \leftarrow \mathbf{a}_i \backslash t_k$, $T_i^R \leftarrow T_i^R \backslash t_k$ //*The unique task in* $T_i^{R'}$ *is selected to remove from task list and candidate removal tasks set* T_i^R
  else if $|T_i^{R'}| > 1$ **then**
    $t_k = \arg\max_{k=1}^{|T_i^{R'}|} \{w_{i,k}^{\ominus\diamond} - w_{i,k}^{\ominus}\}, t_k \in T_i^{R'}$ //*The* t_k *with maximum difference between the global winning RPI and local RPI is selected*
    $\mathbf{a}_i \leftarrow \mathbf{a}_i \backslash t_k$, $T_i^R \leftarrow T_i^R \backslash t_k$ //t_k *is selected to remove from task list and candidate removal tasks set* T_i^R
  end if
  Update R_i^U, $W_i^{\ominus}$, $W_i^{\ominus\diamond}$, $\mho_i^{\diamond}$, $\boldsymbol{\beta}_i$ and τ_{ij} for all tasks in $\mathbf{a}_i$
end while
$\mathbf{a}_i \leftarrow \mathbf{a}_i \cup T_i^R$ // The remaining tasks are put back into $\mathbf{a}_i$ again

Since every UAV locally builds its own task list, two or more UAVs may include the same task. Hence, a conflict resolution phase is demonstrated in Algorithm 3 to resolve the conflict. Take the outputs of task inclusion phase as the inputs and a conflict-free solution is obtained as output in this phase.

- Step 1 (line 1) After communication procedure, the consensus of the global winning UAV list $\boldsymbol{\beta}_i$, global winning RPI list $W_i^{\Theta\diamond}$ and global winning contribution proportion list $\mathfrak{V}_i^{\diamond}$ have been achieved and are now ready to determine if any tasks in the task list should be removed. The candidate tasks set T_i^R to remove from the task list $\mathbf{a}_i$ is determined by Eq. (13):

$$T_i^R = \{\mathbf{a}_i | \boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i\} \tag{13}$$

  In detail, each UAV v_i checks the winning UAVs of all assigned tasks, and the tasks whose winning UAV is not v_i itself, i.e., $\boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i$, are regarded as the conflict tasks.
- Step 2 (lines 3–7) Since a higher contribution proportion implies a better assignment, the UAV with a lower contribution proportion for the conflict task should remove the task from its task list. The globally identical winning contribution proportion list $\mathfrak{V}_i^{\diamond} = [\eta_{i1}^{\diamond}, \ldots, \eta_{iN_t}^{\diamond}]$ are iteratively compared with the local contribution proportion list $\mathfrak{V}_i = [\eta_{i1}, \ldots, \eta_{iN_t}]$. For all tasks in T_i^R, the criterion Eq. (14) is computed:

$$h = \max_{k=1}^{|T_i^R|} \{\eta_{ik}^{\diamond} - \eta_{ik}\} \tag{14}$$

  here $h > 0$ means that reallocating t_k from v_i to the corresponding UAV winner $\beta_{i,k}$ can improve the contribution proportion of task t_k, i.e., the winner $\beta_{i,k}$ can provider more required resources by t_k than UAV v_i. Hence, if $h < 0$ there are no tasks can be released to provide more resources to the tasks in T_i^R and the conflict resolution phase ends.
- Step 3 (lines 8–13) If there is only one task with h, then the task is selected to be removed from the task list $\mathbf{a}_i$ and T_i^R (line 9), and goes to Step 4. Otherwise, when there are two or more tasks with same h, a temporary conflict tasks set $T_i^{R'}$ is defined to stores the tasks with same h, and goes to Step 3 to further select task to delete.

$$T_i^{R'} = [t_k | \arg h, t_k \in T_i^R] \tag{15}$$

  In order to select a specific task with the greatest objective improvement when the contribution proportions are same for multiple tasks, the global winning RPI list $W_i^{\Theta\diamond} = [w_{i,1}^{\Theta\diamond}, \ldots, w_{i,N_t}^{\Theta\diamond}]$ are iteratively compared with the local RPI list $W_i^{\Theta} = [w_{i,1}^{\Theta}, \ldots, w_{i,N_t}^{\Theta}]$. Since a higher RPI means executing the task can achieve higher scores for overall multi-UAV systems, the UAV with the lower RPI for the conflict task should remove it. Similarly, Eq. (16) is computed for all tasks in $T_i^{R'}$.

$$z = \max_{k=1}^{|T_i^{R'}|} \{w_{i,k}^{\Theta\diamond} - w_{i,k}^{\Theta}\} \tag{16}$$

The task with z is selected to remove from the task list $\mathbf{a}_i$ and T_i^R.

- Step 4 (lines 14–16) After removing the selected task, the resources of UAV and time costs for all followed tasks in the task list are updated, and local RPI list $W_i^{\ominus}$, global winning RPI list $W_i^{\ominus\diamond}$ local contribution proportion list $\mathfrak{O}_i$, global winning contribution proportion list $\mathfrak{O}_i^{\diamond}$ and global winning UAV list $\boldsymbol{\beta}_i$ are also updated. Repeating the Step 2 - Step 3 until Eq. (16) is not satisfied or T_i^R is empty. The remaining tasks in T_i^R (if they do exist) are put back into the task list $\mathbf{a}_i$ again. In brief, after repeating the task inclusion phase and conflict resolution phase, the winning UAVs for all tasks have been found. For the task whose winning contribution proportion is equal to 1, its winner UAV can accomplish it individually. It is noted that, the winner for the tasks that cannot be completed by a single UAV denotes the UAV that can provide the most resources for the task or the UAV with maximum global score improvement (z) as Eq. (16) based on maximum contribution proportion.

4.2 Two-stage coalition formation method

For the tasks that cannot be accomplished by a single UAV, i.e., $T_C = \{t_k | 0 < \mathfrak{O}_{i,k}^{\diamond} < 1\}$, the UAV which can provide the most resources has been found as the winning UAV by repeating the task inclusion phase and conflict resolution phase. However, it is still necessary to try to form a coalition to provide enough resources for executing the tasks in T_C.

To this end, a two-stage coalition formation method is designed to form a feasible coalition for the tasks that cannot be completed by a single UAV. In the first stage, the UAVs that can provide the required resources and do not violate the deadline constraints become preliminary coalition members. The coalition size is further reduced in the second stage to obtain the final coalition. The conflict-free solution of iterating the previous two phases is set as the inputs of this two-stage coalition formation method, while the final assignment for all UAVs, coalition and actual start times for all tasks are obtained as the outputs. The details are described as Algorithm 4 and Fig. 3.

4.2.1 Stage 1: select preliminary coalition members

The final winning UAV for the task that cannot be completed by a single UAV is regarded as the leader of the coalition for the task, and the leader further selects members for this coalition.

(1) Idle time slot mechanism (ITSM)

For the tasks which need to be completed by a coalition, the changed time costs for these tasks may have a great impact on the time costs of the following tasks. Hence, it is only allowed to affect the tasks which can be accomplished by a single UAV but do not disturb the tasks which must be completed by a coalition in this paper. To this

Algorithm 4 Two-stage Coalition Formation Method

Input: Tasks lists $\mathbf{a}_i$, local RPI list $W_i^{\ominus}$, global winning UAV list $\boldsymbol{\beta}_i$, global winning contribution proportion list $\mho_i^{\diamond}$ and remaining resources R_i^U for all UAVs

Output: Tasks lists $\mathbf{a}_i$, remaining resources R_i^U for all UAVs, actual start times $\boldsymbol{\tau}$ and final coalition C_k for all tasks

1: **for** t_k with $0 < \mho_{i,k}^{\diamond} < 1$ **do**
2: // **Stage 1 : lines 3 - 13**
3: Compute $\Delta_{L,k}$ by Eq. (17) for leader UAV $v_L = \beta_{ik}$ // *Compute feasible time slot for leader by ITSM*
4: v_L transmits $\Delta_{L,k}$, P_k and $R_k^{T\triangle}$ to neighbors v_j, where $g_{i,j}(t) = 1$ // *Transmit information to neighbors*
5: **for** each v_j, where $g_{i,j}(t) = 1$ **do**
6: **if** $\zeta_{jk} > 0$ and $\mathbf{a}_j^{\oplus_l t_k}$ is feasible // *v_i can provide resources for t_k and there is a position in v_i's task list to insert t_k* **then**
7: Compute $\Delta_{j,k}$ by Eq. (17) // *Compute feasible time slot for v_i to insert t_k*
8: **if** $\Delta_{j,k} \cap \Delta_{L,k} \neq \emptyset$ // *If v_j's feasible time slot overlaps with the leader's* **then**
9: $C_k^P = C_k^P \cup v_j$ // *Include v_j into the preliminary coalition C_k^P*
10: v_j sends $\Delta_{j,k}$ and R_j^U to v_L // *v_j transmits the feasible time slot to leader*
11: **end if**
12: **end if**
13: **end for**
14: // **Stage 2 : lines15 - 37**
15: v_L sorts C_k^p // *Sorts the UAVs in the preliminary coalition*
16: $C_k \leftarrow v_L$ // *Include the leader v_L into the final coalition C_k*
17: $\Delta_k = \Delta_{L,k}$ // *Set the current feasible time slot for final coalition as the leader's feasible time slot*
18: **for** $v_p \in C_k^p, p : 1 \rightarrow |C_k^p|$ // *For each UAV in the preliminary coalition* **do**
19: **if** $\Delta_{p,k} \cap \Delta_k \neq \emptyset$ // *If v_p's feasible time slot overlaps with the current coalition's* **then**
20: $\Delta_k = \Delta_{p,k} \cap \Delta_k$ // *Update the intersection as the feasible time slot of final coalition*
21: $C_k = C_k \cup v_p$ // *Include v_p into the final coalition*
22: **if** $\zeta_{pk} = 1$ // *If v_p can provide all remaining resources that t_k required* **then**
23: **break** // *End the Stage 2 and the final coalition is obtained*
24: **else**
25: Update $R_k^{T\triangle}$ // *Update the remaining resources that t_k still required*
26: $p = p + 1$ // *Continue to include the next preliminary coalition member into the final coalition until the resources that t_k required can be met*
27: **end if**
28: **end if**
29: **end for**
30: **if** $\sum_{i=1}^{|C_k|} R_i^U < R_k^T$ // *If all final coalition members cannot provided enough resources for t_k* **then**
31: Set free all UAVs in C_k // *Release all UAVs in current coalition*
32: **else**
33: **for** $v_p \in C_k, p : 1 \rightarrow |C_k|$ // *For each UAV in the final coalition* **do**
34: $\mathbf{a}_p = \mathbf{a}_p \cup t_k$, $\tau_k = \Delta_k(1) = ES_k$ // *Add t_k and set start times as the earliest start time of the current feasible time slot*
35: Update $\mathbf{a}_p$ and τ // *Update the task list and start times for other tasks*
36: **end for**
37: **end if**
38: **end for**

end, we propose an idle time slot mechanism (ITSM) to compute the feasible time slot $\Delta_{i,n} = [ES_{in}, LS_{in}]$ for v_i to start task t_n as Eq. (17),where t_n needs to be accomplished by a coalition and inserted into the $k+1$ position of $\mathbf{a}_i$ (line 3). ES_{in} and LS_{in} denote the earliest start time and latest start time for v_i to start t_n, respectively.

The earliest start time ES_{in} is computed as the arrival time for the UAV to arrive task, which means that the UAV starts the task as soon as the UAV arrives. For the latest start time LS_{in}, we consider two cases according to the insertion position of task t_n and whether there is a subsequent coalition task. When all subsequent tasks can be completed by a single UAV, LS_{in} calculates as the ES_{in} plus the redundancy time which is the difference between the deadline for starting the last task and the actual start time of the last task. If there is a subsequent task that needs to be completed by a coalition, the computation of feasible time slot is shown as Fig. 4. The start times of tasks which can be performed by a single UAV and between t_k and t_c are delayed. t_c is the first coalition task after insertion position $k+1$. In addition, the waiting time $d^W_{i,c}$ of t_c are also be utilized to create a long feasible time slot. Therefore, t_n can be inserted into $k+1$ position of $\mathbf{a}_i$ without affecting the start times of coalition tasks. It is noted that, the tasks between t_k and t_c all can be performed by a single UAV. LS_{in} should not be later than t_n's deadline s_n.

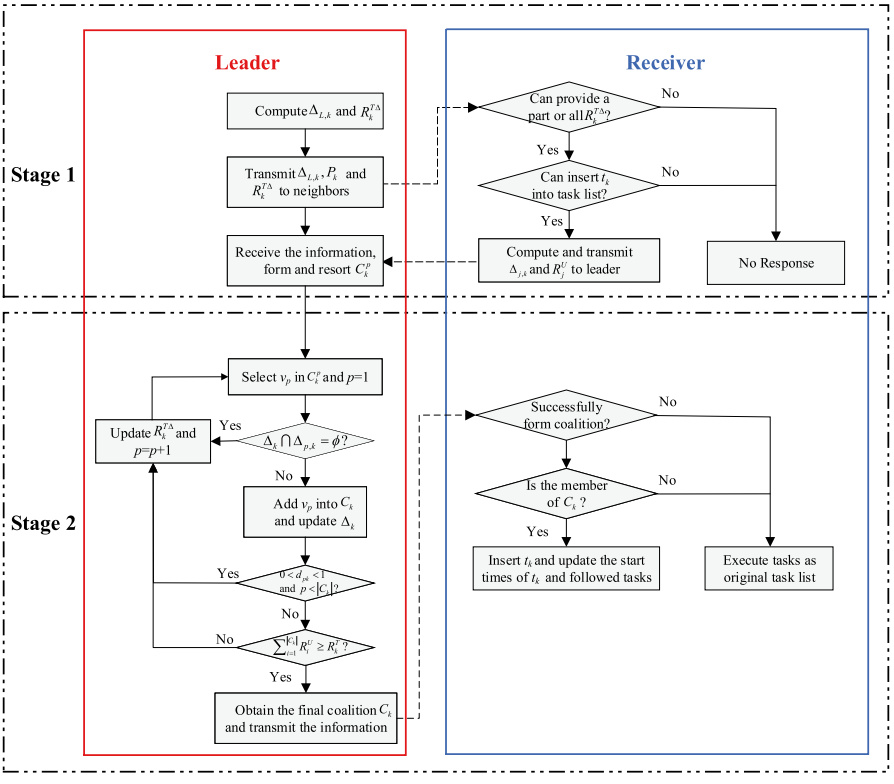

Fig. 3 Flow chart of the proposed two-stage coalition formation method

In conclusion, the feasible time slot $\Delta_{i,n} = [ES_{in}, LS_{in}]$ for v_i start t_n is computed as Eq. (17):

$$ES_{in} = \tau_k + d^E_{i,k} + d^T_{i,k,n} = \tau_{in}$$
$$LS_{in} = \begin{cases} \min ES_{in} + d^W_{i,c} + d^T_{i,k,k+1} - d^T_{i,k,n} - d^E_{i,n} - d^T_{i,n,k+1}, s_n\}, \ 0 < \mho^{\diamond}_{i,c} < 1 \\ \min ES_{in} + (s_{|\mathbf{a}_i|} - \tau_{L|\mathbf{a}_i|}), s_n\}, t_c = \emptyset \end{cases} \tag{17}$$

where $s_{|\mathbf{a}_i|}$ and $\tau_{i|\mathbf{a}_i|}$ represent the deadline and actual start time for the last task in task list $\mathbf{a}_i$ respectively. If the new task is inserted into the last position of the task list, the latest start time is equal to the earliest start time, i.e., $ES_{in} = LS_{in} = \tau_{in}$.

(2) Procedure of stage 1

The leader UAV v_L first computes the feasible start time slot for task t_n by Eq. (17), where v_L cannot perform t_n individually. It is noted that the execution order of t_n in the task list $\mathbf{a}_L$ has been obtained by repeating the task inclusion phase and conflict resolution phase. Then, the leader v_L communicates the information, including the location P_n, the remaining resources required $R^{T\triangle}_n = R^T_n - R^U_L$ and the feasible start time slots $\Delta_{L,n}$ for task t_n, with neighboring UAVs where a communication link exists (line 4). The receiver v_j evaluates whether it can provide a part or all resources required based on the current task list. Here, we define a resource contribution degree ζ_{jn} as Eqs. (18) and (19) to represent the proportion of the resources that UAV v_j can provide the resources currently required to process t_n.

$$\zeta_{jn} = \frac{\sum_{m=1}^{M} D^m_j}{\sum_{m=1}^{M} R^{T\triangle}_{nm}} \tag{18}$$

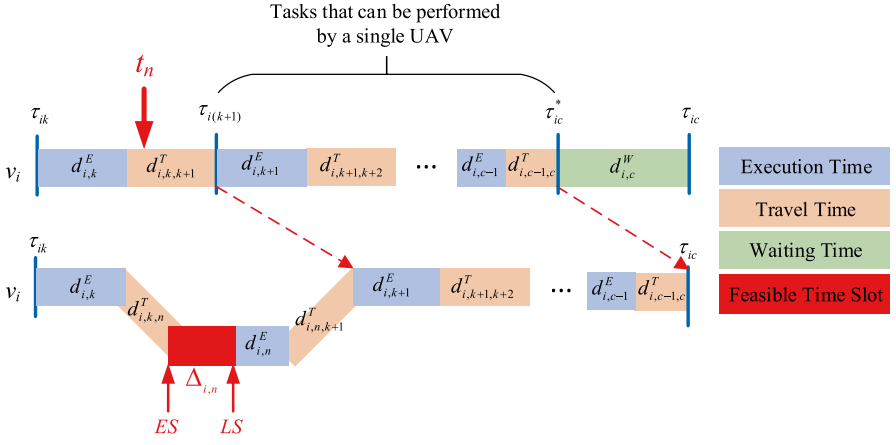

Fig. 4 Illustration of computing the feasible time slot by ITSM

$$D_j^m = \begin{cases} R_{jm}^U, & R_{jm}^U \le R_{nm}^{T\triangle} \\ R_{nm}^{T\triangle}, & others \end{cases} \tag{19}$$

where $R_{nm}^{T\triangle}$ denotes the number of type m resources that t_n still required based on existing coalition members rather than the original required resources R_{nm}^T of t_n before the allocation process. It is noted that, R_j^T always updates as Eq. (20) in the process of task allocation rather than remaining same, to reflect the current resources status of t_j.

$$R_{nm}^{T\triangle} = R_{nm}^T - \sum_{i=1}^{|C_n|} D_j^m \tag{20}$$

If $\zeta_{jn} > 0$, then try to temporarily insert t_n into all positions of $\mathbf{a}_j$ by task inclusion phase. If there is a position available to insert t_n, then compute the feasible start time slot $\Delta_{j,n} = [ES_{jn}, LS_{jn}]$ by Eq. (17) (line 7). If the $\Delta_{j,n}$ and $\Delta_{L,n}$ overlap, i.e., $\boldsymbol{\Delta}'_{\boldsymbol{j,L,n}} = \Delta_{j,n} \bigcap \Delta_{L,n} \neq 0$, then receiver v_j becomes one of the members of the preliminary coalition C_n^P, and sends the information of feasible start time slot $\Delta_{j,n}$ and the resources it still carried R_j^U to the leader v_L (lines 9-10). It means that when the receiver has a position that can insert t_n and the feasible start time slot of this position overlaps the feasible start time slot of the leader, it becomes a preliminary coalition member. Otherwise, when all positions cannot provide an overlapped feasible start time with the leader, the UAV is not selected. It is noted that the leader has the ability to calculate the overlapped feasible start time slot $\Delta'_{j,L,n}$ and contribution degree ζ_{jn} based on the received information.

4.2.2 Stage 2: reduce coalition size

The leader v_L receives the information from preliminary coalition members in C_k^p and needs to further reduce the coalition size to obtain the final coalition members. We detail the next three different criteria for the leader to rank UAVs in a preliminary coalition (line 15). Each approach prioritizes the UAVs by a different characteristic and aims towards optimizing according to a different metric.

- **Max $\boldsymbol{\zeta_{jk}}$**: Order preliminary coalition members according to the contribution degree ζ_{jk}. The UAVs that can provide more resources come first. This strategy can reduce the coalition size and fewer UAVs would add to the coalition. In addition, fewer coalition members will result in fewer UAVs may be affected by the task's delay.

- **Max $\Delta'_{j,L,k}$**: UAVs are ordered by overlapped feasible start time slot $\Delta'_{j,L,k}$. Since the feasible start time slot of the UAV must overlap with the leader's, a longer overlapped feasible start time slot means the coalition can start the task within a more flexible time slot.
- **Min ES_{jk}**: Prioritizes UAVs according to their earliest start time (ES_{jk}). Since the delay in the start time of the task that requires a coalition to complete will result in the subsequent tasks of all coalition members being delayed. Hence, the later the task start, the lower reward received for the overall multi-UAV system which can be inferred from Eq. (3). To this end, the UAV with the earliest start time has the highest priority, which can receive a higher total reward.

First, after the leader v_L joins C_k, the final feasible start time slot for t_k is equal to the feasible start time slot of leader UAV, i.e., $\Delta_k = \Delta_{L,k}$ (lines 16-17). Then, iteratively add the UAVs in rearranged preliminary coalition C_k^p in order (lines 18-19). If v_p still has common feasible start time slot with the exist coalition feasible start time slot, i.e., $\Delta_k \bigcap \Delta_{p,k} \neq \emptyset$, the feasible start time for final coalition is updated as the intersection of the two, i.e., $\Delta_k = \Delta_k \bigcap \Delta_{p,k}$, and v_p is included into the final coalition C_k. When v_p cannot provide the resources that t_k still required or $\Delta_k \bigcap \Delta_{p,k} = \emptyset$, the adding process skips v_p and continues to include other UAVs to final coalition. The stage 2 ends when there is a UAV v_p which can provide all remaining required resources of t_k, i.e., $\zeta_{pk} = 1$ or the all UAVs in preliminary coalition cannot provide adequate resources for t_k. The final coalition for completing task t_k is denoted as C_k and the feasible start time for task t_k is defined as the intersection of feasible start time slots for all coalition members as Eq. (21):

$$\Delta_k = \bigcap_{i=1}^{|C_k|} \Delta_{i,k} \neq \emptyset, \forall v_i \in C_k \tag{21}$$

The task is regarded as failed when there are not enough resources provided by all UAVs in the preliminary coalition, and all UAVs in C_k is set free (lines 30–31). When the final coalition can provide all required resources, all final coalition members include the task t_k and set the start time of t_k as the earliest start time in the final feasible time slot. After that, all final coalition members update the start times of all subsequent tasks (lines 33–36).

4.3 Reassignment application in dynamic environment

Algorithm 5 Reassignment Application of the Two-stage Coalition Formation Method

1: v_d detects the new task t_n within its detection radius
2: v_d computes the resource contribution degree ζ_{dn} as Eq. (18)
3: v_d computes $\Delta_{d,n}$ by Eq. (17) for all positions in $\mathbf{a}_d$
4: $\Delta_{d,n} = \max_{l=1}^{|\mathbf{a}_d|+1} |\Delta_{d,n}|$
5: **if** $\Delta_{d,n} \neq \emptyset$ and $\zeta_{dn} = 1$ **then**
6: Allocate t_n to v_d
7: **else**
8: v_d transmits the information of t_n to neighbors U_N
9: **for** each $v_j \in U_N$ **do**
10: Compute the resource contribution degree ζ_{jn} as Eq. (18)
11: **if** $\zeta_{jn} > 0$ **then**
12: Compute $\Delta_{j,n}$ by Eq. (17) for all positions in $\mathbf{a}_j$
13: $\Delta_{j,n} = \max_{l=1}^{|\mathbf{a}_j|} |\Delta_{j,n}|$
14: **if** $\Delta_{j,n} = \emptyset$ **then**
15: $\zeta_{jn} = 0$
16: **end if**
17: **end if**
18: **end for**
19: $v_L \leftarrow \arg\max_{j=1}^{|U_N|} \zeta_{jn}$//Carry communication to elect the UAV with maximum resource contribution degree as the leader
20: **if** $\zeta_{Ln} = 1$ **then**
21: Allocate t_n to v_L
22: Break
23: **else**
24: Carry the two-stage coalition formation method
25: **end if**
26: **end if**

When all UAVs execute tasks as the preallocation results which are computed based on known information of UAVs and tasks, there may be unexpected new tasks introduced to the problem, which may lead to low efficiency or failure of the original plan. With the purpose of reallocating the new tasks within a short time and avoiding unnecessary computation resources, it is preferred to allocate the new task to the UAV which can perform it individually rather than a coalition. To this end, this section describes the ability of the two-stage coalition formation method to address the dynamic task reallocation problem.

When a UAV v_d detects a new task t_n in the process of executing the tasks in its task list, it first determines whether it can provide all required resources for performing t_n. Furthermore, determine whether exists a position to insert t_k without affecting the time costs of the assigned tasks which need to be completed by a coalition (line 5). If v_d can, then it includes t_k into its task list and the reassignment ends (line 6). However, if v_d cannot perform t_k individually, it transmits the information of t_n to all its neighbors U_N, and each neighboring UAV v_j computes the resource contribution degree ζ_{jn} by Eq. (18) (line 10). If v_j can provide resources t_n required, i.e., $\zeta_{jn} > 0$, the feasible start time slots for all positions are computed by Eq. (17) and the widest one is selected as the final feasible start time slot $\Delta_{j,n}$ (lines 12–13). If there is no

position to insert t_n, we set the resource contribution degree as 0 (lines 14–15). After all, neighbors have computed the resource contribution degrees and feasible start time slots, a simple communication is triggered to elect the UAV which can provide the most resources for r_n as the leader (line 19). When the leader can provide all resources t_n required individually, t_n is allocated to it (lines 20–22). Otherwise, carry the two-stage coalition formation method to further form a coalition for the new task t_n (line 24).

Remark 1 If the UAV can provide required resources for multiple tasks as the leader or receiver, there may exist resource conflicts or deadlock. In this paper, we consider two types of resource deadlock, a UAV can provide resources for multiple tasks and a leader can provide resources for multiple tasks, and both of these deadlocks can be solved by comparing the static value SV_j of the task. In detail, since a higher static value means that the task is more important and with higher priority, the UAV evaluates the available resources in the order in which the static value decreases. It means that only when the coalition formation process of the task with the higher static value is completed, the coalition formation stage of the task with the lower static value can be carried out.

4.4 Computational complexity

To asses the computational complexity of running RCPIA on one UAV, the method used in [43] is followed. According to the structure of RCPIA, the analysis of computational complexity is divided into three parts: (1) task inclusion phase and conflict resolution phase part, (2) two-stage coalition formation part, and (3) reassignment part.

(1) For the first part, the major computational complexity arises from the calculation of RPI and IPI values and the updated scores for the remaining tasks in the task list. In task inclusion phase, a maximum computational complexity of $(|\mathbf{a}_i|+1)(|\mathbf{a}_i|+2)M_1\sigma/2+|\mathbf{a}_i|\sigma$ is requested to compute the IPI and update the scores for remaining tasks when inserting a new task into $\mathbf{a}_i$. $|\mathbf{a}_i|$ denotes the cardinality of the task list $\mathbf{a}_i$. M_1 denotes the number of tasks that are not yet in $\mathbf{a}_i$ and meet the constraints in Eq. (4). σ implies the complexity of computing the local score of a task. For the conflict resolution phase, as most operations are simple rule-based logical judgments, few computations are required for consensus. It is assumed that a total of M_2 tasks are intended to be removed from $\mathbf{a}_i$. This will requires $|\mathbf{a}_i|M_2\sigma-M_2(M_2+1)\sigma/2+(|\mathbf{a}_i|-1)\sigma$ computational complexity maximumly. It can be inferred that the major computational complexity is dominated by the task inclusion phase and defined as $O((m_i-|\mathbf{a}_i|)|\mathbf{a}_i|^2M_1\sigma N_u)$ at each iteration to compute IPI and RPI. $m_i-|\mathbf{a}_i|$ is the maximum number of tasks that can be added into a UAV's task list during each iteration of the algorithm. It is noted that, the difference between baseline PI and RCPIA lies on the computation and comparison of contribution proportion. In addition, the changes in the contribution ratio of one task will not affect the contribution

ratio of other tasks in the task list as shown in Eq. (6). Define ξ as the computation time of contribution degree for a position in the task list. Hence, the maximum computational complexity of contribution degree in this two phases is $O(M_1 N_u(m_i - |\mathbf{a}_i|)\xi)$. To sum up, the computational complexity of these two phases are $O((m_1 - |\mathbf{a}_i|)(\xi + |\mathbf{a}_i|^2\sigma)N_u M_1)$.

(2) For the second part, it is assumed that up to N_u UAVs participate in the coalition construction for performing the task t_j which needs N-type of resources. Define κ as the computation time of idle time slot of a position in the task list. In the first stage, by calculating the idle time slot for all UAVs by ITSM, the preliminary coalition members are selected and the computational complexity is $O((|\mathbf{a}_i + 1|)N_u N\kappa)$. In the second stage, the preliminary members are first sorted by one of the three criteria (discussed in Section 4.2.2) which result in $O(N_u lgN_u)$ computational complexity maximally. After that, every UAV must be iteratively checked until the resource constraints of tasks are met. Hence, the maximum computational complexity of the second stage is $O(N_u N + N_u lgN_u)$. In conclusion, the computational complexity of the two-stage coalition formation method is $O(N_u(lgN_u + N + (|\mathbf{a}_i| + 1)\kappa N))$.

(3) For the third part, the detection UAV first computes the contribution degree and idle time slots for all positions in $\mathbf{a}_i$ which requires $O(\xi + (|\mathbf{a}_i| + 1)\kappa)$. Then, every UAV needs to compute the feasible time slot for all positions in its task list and contribution degree, which needs $O(N_u(\xi + (|\mathbf{a}_i| + 1)\kappa))$ computational complexity maximally. In summary, the reassignment part requires $O((N_u + 1)((|\mathbf{a}_i| + 1)\kappa + \xi)$.

It can be concluded that the main computational complexity is still dominated by the calculation of IPI and RPI. The computational complexity of the two-stage coalition formation method and its reassignment application are all related to the number of available positions $|\mathbf{a}_i| + 1$ and the number of UAVs N_u, which means that these two parts are all polynomial-time algorithms that scale well with the number of the UAVs and tasks in its task list. Hence, there is not too much computing time consumed in these two parts.

5 Numerical results

This section presents the simulation results of RCPIA to demonstrate its effectiveness and performance. The test scenario is first introduced and used to illustrate the solving process of RCPIA, including the reassignment application for dealing with the new emerging task. In addition, RDAATSM is conducted to compare with RCPIA to demonstrate its efficiency.

5.1 Test scenario

To test the performance of RCPIA, the combat scenarios mentioned in [15] and [16] are taken as references to design a combat scenario shown in Fig. 5. There are six

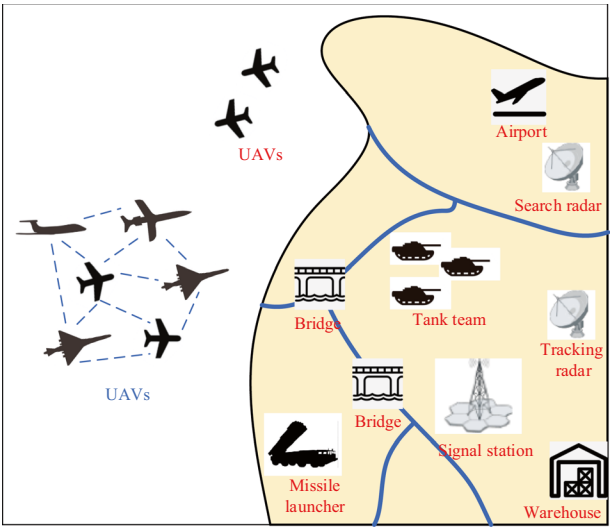

Fig. 5 Simulation example of combat situation

Table 3 Resources capabilities of UAVs

UAVs	ATAM/R_1	AGM/R_2	Aerial Boom/R_3	BGL/R_4	Aircraft Gun/R_5	EMI/R_6	Velocity
v_1	2	3	3	2	200	0	250
v_2	2	6	0	2	550	0	280
v_3	1	3	6	2	500	1	230
v_4	7	0	3	3	200	1	280
v_5	2	0	6	4	450	1	400
v_6	0	6	3	3	300	0	320

heterogeneous UAVs, 16 tasks and six types of resources in this considered scenario. The resources capabilities of UAVs and the resource requirements for attack tasks are displayed in Tables 3 and 4 respectively, where the unit of time is an hour and the positions of all UAVs are initialized as (20,30,4). The commander can set the deadlines for attack tasks according to the battlefield situation to avoid further deterioration of the situation. Also, the deadlines can be used for the commander to control the different stages of the combat. In addition to EMI, other resources will be continuously consumed when performing tasks.

It is assumed that all UAVs perform tasks at an altitude of 4 *km*. Note that the scenario settings described in this paper are not necessary for the algorithm to work. All methods are conducted on MATLAB R2016a with AMD Ryzen 7 PRO CPU @ 1.70 GHz.

Table 4 Resource requirements for attack tasks

Tasks	ATAM/R_1	AGM/R_2	Aerial Boom/R_3	BGL/R_4	Aircraft Gun/R_5	EMI/R_6	Processing time	Position	Deadline	Static Value
t_1/Tank Team1	0	2	0	0	0	0	0.5	(70,50,0)	4	60
t_2/Tank Team2	0	0	3	1	0	1	0.8	(70,80,0)	3.5	70
t_3/Tank Team3	0	1	0	0	100	1	0.3	(80,100,0)	3.8	65
t_4/Bridge1	0	2	0	0	300	0	1.2	(60,40,0)	5	40
t_5/Bridge2	0	3	0	2	0	0	1	(65,25,0)	4.2	80
t_6/Bridge3	0	1	3	1	0	0	1.5	(90,70,0)	5	75
t_7/Radar1	0	0	3	1	0	1	0.3	(75,30,0)	5.5	55
t_8/Radar2	0	0	0	1	0	1	0.4	(80,75,0)	4.4	65
t_9/Radar3	0	0	0	1	0	1	0.4	(65,15,0)	4.5	65
t_{10}/Warehouse	0	0	3	0	0	1	1	(95,15,0)	5	70
t_{11}/Airport	0	2	0	2	100	1	2	(90,90,0)	4	85
t_{12}/Information Station	0	1	0	1	0	1	0.5	(70,40,0)	3.4	80
t_{13}/UAV1	1	0	0	0	200	1	1	(70,50,4)	4.8	60
t_{14}/UAV2	1	0	0	0	100	1	0.8	(70,50,4)	4.8	60
t_{15}/UAV3	0	0	0	0	200	1	0.3	(70,50,4)	4.8	65
t_{16}/Missile launcher	0	2	3	1	0	0	0.2	(65,50,0)	4.5	90

5.2 The feasibility of RCPIA

5.2.1 Task assignment for known tasks

The allocation results of known tasks derived from RCPIA are shown in Table 5. The task sequence for each UAV and the actual start times are displayed where the coalition tasks are in bold. Remaining resources for all UAVs are also displayed. It can be seen that all tasks are allocated by RCPIA. Furthermore, it is noted that 2 tasks (t_7 and t_{11}) need to be cooperatively performed by a coalition at the same time displayed in Table 6. The leader for each coalition tasks are in bold. To clarity, the schedules for all UAVs are also demonstrated in Fig. 6 and the coalition task is all in red where tasks that can be performed by a single UAV are in gray.

Since RCPIA prefers to allocate the tasks to the UAV which can individually complete them, only two tasks (t_7 and t_{11}) whose leaders (v_3 and v_2) cannot provide enough resources after iterating task inclusion phase and conflict resolution phase. Hence, the leaders form coalition for tasks by the two-stage coalition formation method. It can be seen from Table 6 that v_3 as the leader of t_7 first selects v_1 and v_5 as the preliminary members and the two UAVs provide same number of resources for t_7. Since the longer feasible time slot of v_5 is overlapped with v_3's, the v_5 is selected as the final coalition members to cooperatively perform t_7 and it needs to wait 0.2066h until v_3 arrives. The two UAVs start t_7 at 0.7703h. Similarly, as the leader of t_{11}, v_2 first chooses v_4 and v_5 as the preliminary coalition members by the phase 1. The two preliminary members both provide 0.0095 contribution degree for t_{11}. Finally, the leader v_2 chooses v_4 as the final coalition members to start t_{11} at 1.4613h.

Therefore, this section demonstrates the fact that most of the tasks are allocated to a single UAV by RCPIA rather than a coalition. In addition, RCPIA can further make full use of resources to cooperatively form coalitions for the tasks that cannot be completed by a single UAV.

Table 5 Assignment results for known tasks

UAVs	$\mathbf{A}_0(\boldsymbol{\tau}_0)$					Remaining Resources
	$\mathbf{a}_{i,1}(\boldsymbol{\tau}_{i1})$	$\mathbf{a}_{i,2}(\boldsymbol{\tau}_{i2})$	$\mathbf{a}_{i,3}(\boldsymbol{\tau}_{i3})$	$\mathbf{a}_{i,4}(\boldsymbol{\tau}_{i4})$	$\mathbf{a}_{i,5}(\boldsymbol{\tau}_{i5})$	
v_1	14(0.2154)	6(1.1285)				[1,2,0,1,100,0]
v_2	3(0.3293)	1(0.8087)	**11(1.4613)**			[2,1,0,0,350,0]
v_3	12(0.2217)	**7(0.7703)**	2(1.2888)	13(2.2192)	4(3.2807)	[0,0,0,0,0,1]
v_4	15(0.1923)	**11(1.4613)**				[5,0,3,0,0,1]
v_5	9(0.1186)	**7(0.7703)**	8(1.1835)	10(1.7381)		[2,0,0,0,250,1]
v_6	16(0.1539)	5(0.5320)				[0,1,0,0,300,0]

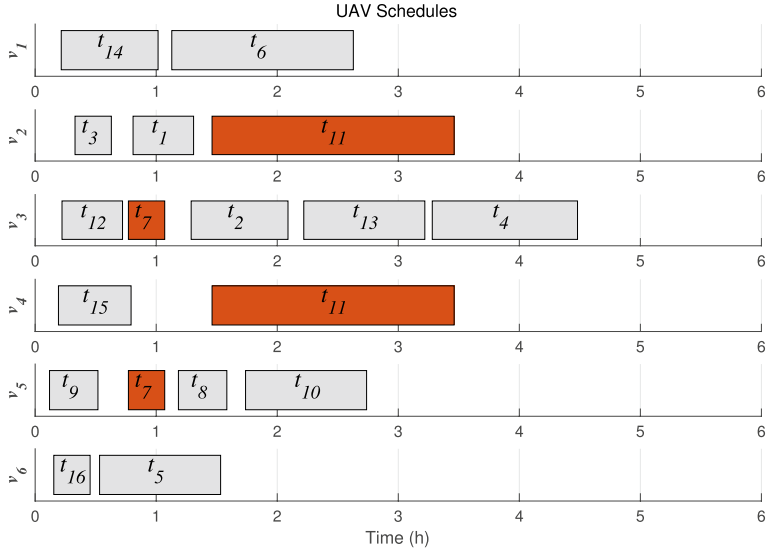

Fig. 6 Schedule of all UAVs for allocating known tasks

Table 6 Coalition formation process of some known tasks

Coalition tasks	Stage	UAV	Feasible start time slot	Provide resources	Contribution degree	Waiting time
t_7	Stage 1 : Preliminary	v_1	[0.2200,4.09]	[0,0,0,1,0,0]	0.2	/
		$\boldsymbol{v_3}$	[0.7703,2.4896]	[0,0,3,0,0,1]	0.8	/
		v_5	[0.5637,4.3358]	[0,0,0,1,0,0]	0.2	/
	Stage 2 : Final	$\boldsymbol{v_3}$	[0.7703.2.4896]	[0,0,3,0,0,1]	0.8	0.2066
		v_5	[0.7703.2.4896]	[0,0,0,1,0,0]	0.2	0
t_{11}	Stage 1 : Preliminary	$\boldsymbol{v_2}$	[1.4613,4.1731]	[0,2,0,2,100,0]	0.9905	/
		v_4	[0.9520,4.1023]	[0,0,0,0,0,1]	0.0095	/
		v_5	[2.9260,4.2013]	[0,0,0,0,0,1]	0.0095	/
	Stage 2 : Final	$\boldsymbol{v_2}$	[1.4613,4.1023]	[0,2,0,2,100,0]	0.9905	0.5093
		v_4	[1.4613,4.1023]	[0,0,0,0,0,1]	0.0095	0

5.2.2 Reassignment for the new emerging task

In a dynamic environment, some new tasks may be introduced into the problem, which leads to low efficiency or failure of the original solution. Hence, to illustrate the feasibility of RCPIA to cope with the new emerging task, set the new emerging task as the new task t_{17} appears at [50,50,0] at 0.3h. The information of new task is displayed in Table 7 and the reassignment of t_{17} is shown in Table 8. The coalition tasks are still in bold while the tasks whose start times are affected are in italics.

Table 7 The resource requirements for attacking the new emerging task t_{17}

Tasks	ATAM/R_1	AGM/R_2	Aerial Boom/R_3	BGL/R_4	Aircraft Gun/R_5	EMI/R_6	Processing time	Position	Deadline	Static Value
t_{17}/UAV 4	2	0	0	0	200	1	0.2	(50,50,0)	3.2	100

Table 8 Reassignment results of all UAVs after a new task appearing

UAVs	$\mathbf{A}_0(\boldsymbol{\tau}_0)$					Remaining Resources
	$\mathbf{a}_{i,1}(\boldsymbol{\tau}_{i1})$	$\mathbf{a}_{i,2}(\boldsymbol{\tau}_{i2})$	$\mathbf{a}_{i,3}(\boldsymbol{\tau}_{i3})$	$\mathbf{a}_{i,4}(\boldsymbol{\tau}_{i4})$	$\mathbf{a}_{i,5}(\boldsymbol{\tau}_{i5})$	
v_1	14(0.2154)	6(1.1285)				[1,2,0,1,100,0]
v_2	3(0.3293)	1(0.8087)	**11(1.4613)**			[2,1,0,0,350,0]
v_3	12(0.2217)	**7(0.7703)**	2(1.2888)	13(2.2192)	4(3.2807)	[0,0,0,0,0,1]
v_4	***17(0.5008)***	*15(0.7722)*	**11(1.4613)**			[3,0,3,0,0,1]
v_5	9(0.1186)	**7(0.7703)**	8(1.1835)	10(1.7381)		[2,0,0,0,250,1]
v_6	16(0.1539)	***17(0.5008)***	*5(0.7919)*			[0,1,0,0,100,0]

It can be seen that the start times of t_{15} and t_5 are delayed compared to the original schedules of v_4 and v_6. To clarity, the schedules for all UAVs after reallocation are shown in Fig. 7, and the allocation of the new task t_{17} is in green and the coalition process for t_{17} is also presented in Table 9.

The reallocation process of t_{17} (Table 9) is described as follows in detail. Since the detection UAV v_2 cannot provide enough resources for the new task t_{17}, it transmits the information of t_{17} to its neighbors through the communication network. After each neighbor computes the contribution degree and feasible time slot, v_6 is selected as the new leader of t_{17}. However, v_6 only can contributes [0,0,0,0,200,0] resources for t_{17} and there are still [2,0,0,0,0,1] resources required. Hence, it selects preliminary coalition members by the first stage of the two-stage coalition formation method. v_1 and v_4 can contribute 0.0049 and 0.0147 contribution degree for t_{17}.

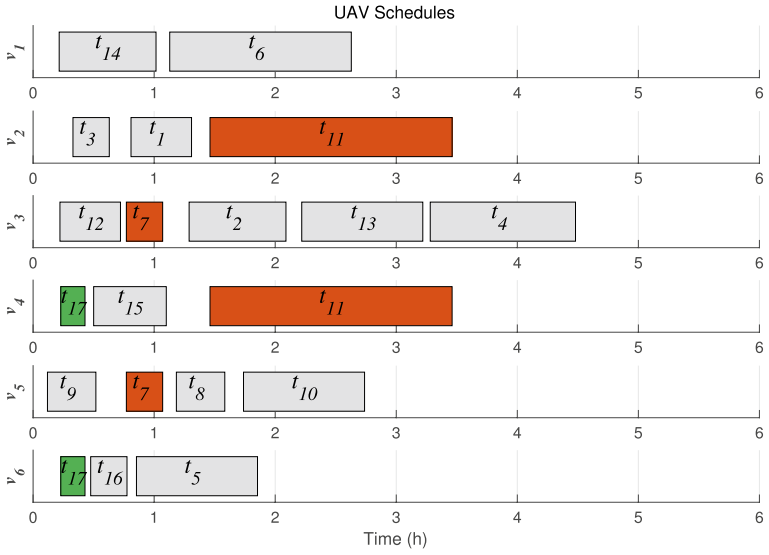

Fig. 7 Scheduler of all UAVs after reallocating the new task

Table 9 Coalition formation process of the new emerging task

Coalition tasks	Stage	UAV	Feasible start time slot	Provide resources	Contribution degree	Waiting time
t_{17}	Stage 1 : Preliminary	v_1	[0.3442,3.2000]	[1,0,0,0,0,0]	0.0049	/
		v_4	[0.3288,0.6302]	[2,0,0,0,0,1]	0.0147	/
		$\boldsymbol{v_6}$	[0.5008,3.2000]	[0,0,0,0,200,0]	0.9853	/
	Stage 2 : Final	v_4	[0.5008,0.6302]	[2,0,0,0,0,1]	0.0147	0.1720
		$\boldsymbol{v_6}$	[0.5008,0.6302]	[0,0,0,0,200,0]	0.9853	0

Finally, the leader v_6 chooses v_4 as the other final coalition member to cooperatively start t_{17} at 0.5008h. It is worth mentioning that, to insert the new task into v_4's task list, RCPIA creates a longer feasible idle time slot and not affects the coalition task t_{11} by delaying the start time of t_{15}. Since t_{15} is only performed by v_4, this delay will not affect the schedules of other UAVs'. In addition, since v_6 can individually complete tasks t_{16} and t_5, the start times of these two tasks can all be delayed to create a much long idle time slot. Hence, the feasible time slot for v_6 to start t_{17} is [0.5008,3.2], where 3.2h is the deadline for starting t_{17}.

This section shows that RCPIA is able to create a longer feasible idle time slot for the new task by shifting the start times of the tasks that can be performed by a single UAV. At the same time, the start times of all coalition tasks will not be affected. This will not only ensure the preallocation will not be greatly affected, but also provides greater flexibility to insert the new task.

5.3 The influence of UAVs number and tasks number

To further analyze the influence of the number of UAVs and tasks on RCPIA more comprehensively and generally, we consider two series of simulations. It is noted that, each setup with the same number of tasks and UAVs but different resources distribution runs 50 times to avoid contingency. Since RDAATSM [13] can also support the simultaneously starting tasks and resource constraint, it is used to be compared with RCPIA in terms of the average percentage of allocating known tasks, the average percentage of tasks to be completed by a UAV coalition to all allocated tasks, the average runtime and the average total score of allocating known tasks.

First, we consider 40 known tasks need to be allocated to UAVs whose number increases from 4 to 16. The comparison results are shown in Fig. 8. As the number of UAVs increases, more tasks can be allocated due to more resources can be provided as shown in Fig. 8a. In addition, RCPIA always allocates more tasks than RDAATSM. Since RDAATSM is developed based on CBBA, it must meet the marginal diminishing characteristics which lead to the tasks with the greatest score being first selected rather than the one with the greatest contribution degree. This may lead to forming more coalitions to provide enough resources for the task. However, more coalitions may indicate higher failure rates since the task is only allowed to insert into the appropriate task idle period without affecting

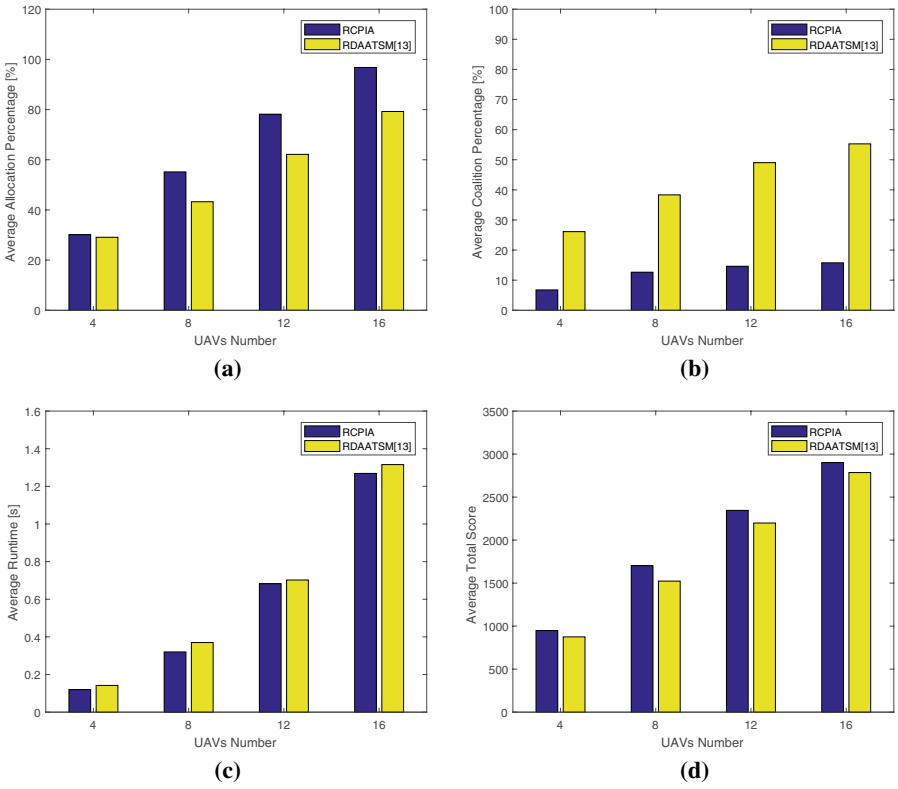

Fig. 8 The simulations are tested for allocating 40 known tasks to an increasing number of UAVs which are 4, 8, 12, 16. Run RCPIA and RDAATSM 50 times and comparison of **a** Average allocation percentage of known tasks, **b** Average percentage of allocated tasks that need to be completed by a coalition, **c** Average runtime, and **d** Average total score are shown

the established plan. Hence, the more coalitions are required to allocate the tasks, the low allocation percentage in RDAATSM caused. For the coalition percentage of allocated tasks shown in Fig. 8b, fewer tasks are allocated to coalitions for RCPIA compared to RDAATSM, which results from RCPIA prefers to allocate the tasks to the UAV that can complete it individually. For the runtime, as the number of UAVs increases, more conflicts will be resolved in the conflict resolution phase as all UAVs build their task lists locally and more time is consumed in the task inclusion phase to re-add tasks after the conflict resolution phase. Hence, as seen in Fig. 8c, the average runtime of both two algorithms increases as more UAVs are introduced into the problem. Besides, RCPIA consumes a little less runtime to obtain the results compared to RDAATSM. The reason is that even though the two-stage coalition formation method consumes more time to form coalitions for the tasks that cannot be performed by a single UAV, the marginal diminishing characteristics of RDAATSM causes releasing all subsequent tasks of added/deleted tasks. This operation may consume more runtime. As more

tasks are allocated by RCPIA, its total scores are higher than that of RDAATSM as shown in Fig. 8d.

Then, we consider the scenario that 20, 40, 60, and 80 known tasks need to be allocated to 8 UAVs. The comparison results are shown in Fig. 9. As the number of tasks increases, fewer tasks are allocated successfully as shown in Fig. 9a since the resources of 8 UAVs are certain and fewer tasks can be provided enough resources. In addition, fewer tasks are assigned to the coalition by RCPIA compared to RDAATSM in Fig. 9b. The reason is that when more tasks need to be assigned, RCPIA prefers to select tasks whose required resources better match those UAVs carry. Furthermore, RDAATSM selects the tasks that can achieve a higher score, which means that more coalitions are needed. As seen in Fig. 9c, as the increasing number of tasks need to be assigned, there is increasing runtime consumed due to calculating the IPI/RPI and bids in RCPIA and RDAATSM respectively. Similarly, fewer coalition tasks result in less time taken in forming a coalition by the proposed two-stage coalition formation method in RCPIA, while more runtime is required in

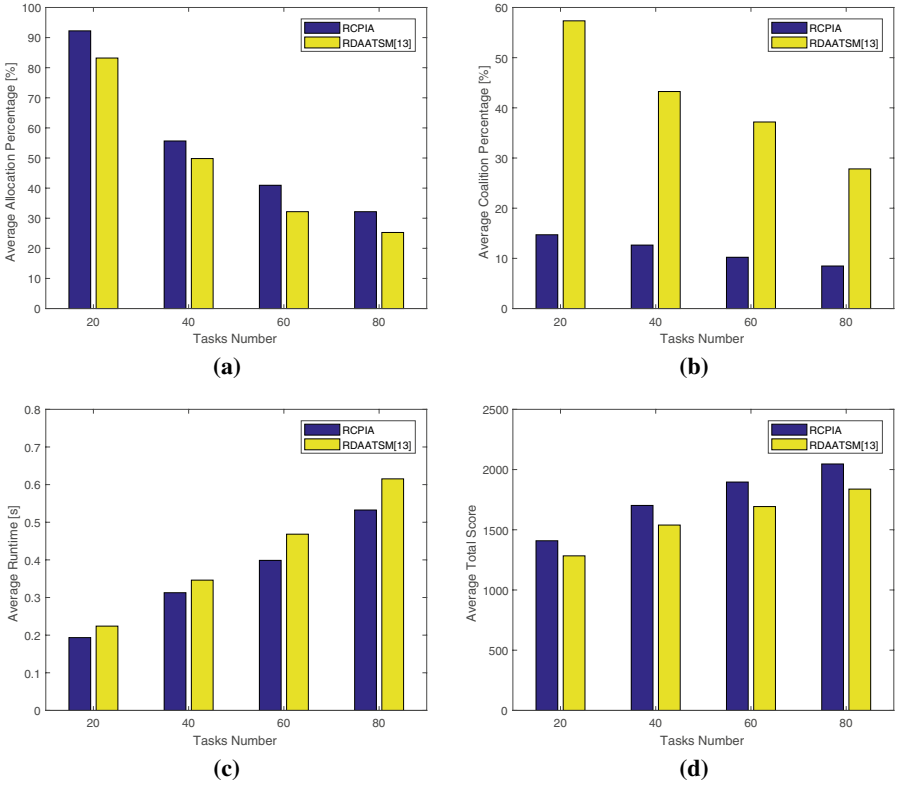

Fig. 9 The simulations are tested for an increasing number of known tasks which are 20, 40, 60, and 80 need to be allocated to 8 UAVs. Run RCPIA and RDAATSM 50 times and comparison of **a** Average allocation percentage of known tasks, **b** Average percentage of allocated tasks that need to be completed by a coalition, **c** Average runtime, and **d** Average total score are shown

RDAATSM to form a UAV squad to provide enough resources for more coalition tasks. Fewer allocated tasks of RDAATSM imply lower total scores as shown in Fig. 9d.

In conclusion, the characteristic of preferring to allocate the tasks to the UAV which can complete them individually in RCPIA avoids some unnecessary cooperation to form a coalition. This not only leads to more tasks are allocated and fewer tasks are allocated to a coalition, but also higher total scores are obtained. In addition, although the two-stage coalition formation method consumes much time to form a coalition, fewer tasks need to form coalitions, so less average runtime is spent.

5.4 The performance of RCPIA in reallocating the new task

To further demonstrate the performance of RCPIA in dealing with the new emerging task, this section considers three different appearance times of the new task, i.e., early, middle and late time which defined as the 10 percent, 40 percent and 80 percentage of the latest task completion time of the preallocation schedules. For a given 6 UAVs and 20 known tasks, each setup runs 50 times and the average results in terms of the success rate of reallocation, the percentage of a coalition, the consumed runtime and reallocation score of reallocating the new task successfully are displayed in Table 10.

It can be seen that, with the delay of the emergence of the new task, fewer simulations can reallocate the new task successfully as shown in Table 10, which results from that fewer positions left to insert the new task. In addition, RCPIA always can allocate the new task in more simulations compared with RDAATSM. This is because that most of the preallocated tasks can be completed by a single UAV in RCPIA and the start times of these tasks can be flexibly shifted to create a long feasible time slot to insert the new task. Meanwhile, since RDAATSM only allows to insert the new task into the synchronization wait time and the start times of preallocated tasks cannot be affected, fewer simulations can allocate the new task successfully, i.e., the average success rate of reallocation for RCPIA is higher than RDAATSM. In addition, the average coalition percentage of RCPIA is lower than RDAATSM since RCPIA prefers to allocate the tasks to the UAV that can complete

Table 10 Performance comparison when assigning the new task with different appearance time

Appearance Time	Success rate of reallocation		Percentage of coalition		Runtime		Reallocation score	
	RCPIA	RDAATSM	RCPIA	RDAATSM	RCPIA	RDAATSM	RCPIA	RDAATSM
Early Time	78.23%	52.85%	62.07%	73.21%	0.1564s	0.2703s	1140.5	1238.4
Middle Time	64.38%	43.67%	65.29%	75.43%	0.1203s	0.2210s	1138.1	1293.7
Late Time	56.17%	37.89%	66.68%	77.28%	0.0927s	0.1517s	1129.3	1189.6

tasks individually. It must be noted that, since the appearance time and the start times of coalition tasks are random and not influenced by the appearance time but related to the preallocated schedules, the coalition percentage of the two algorithms does not show a obvious change trend. As for the runtime, since RCPIA prefers to allocate the tasks to the UAV that can perform the tasks individually, less computation time is spent in forming a coalition for the new task, i.e., the runtime of RCPIA reallocating the new task is less than RDAATSM. Besides, since the number of feasible insertion positions decreases as the appearance time increases, less computation time is required to calculate the IPI/RPI or bids for the two algorithms respectively, i.e., the average runtime for allocating the new task decreases. For the average reallocation score of the simulations which reallocate the new task successfully, RDAATSM performs better than RCPIA since it allocates the new task to the UAV with the highest bid.

In conclusion, most allocated known tasks are performed by a single UAV in RCPIA, which results that the preallocation is more flexible to cope with the new task. Hence, RCPIA can allocate the new emerging task in more simulations and less runtime is consumed compared to RDAATSM, which demonstrates the performance of RCPIA in reallocating the new task in dynamic environment.

6 Conclusion and future works

The task assignment problem of a multi-UAV system with limited resources is one of the most extensive research domains and many efforts have been made. Although the existing algorithm considers forming a coalition to accomplish tasks, which require a different number of resources, they cannot guarantee that the coalition UAVs reach the task position at the same time. In addition, the algorithm should be more flexible and robust when encountering new emerging tasks. To this end, this paper proposes a distributed heuristic algorithm called RCPIA to address the task allocation problem for heterogeneous UAVs with limited resources. First, the task allocation problem with the constraints of resources and simultaneously starting tasks is modeled in the mathematical formulation. Second, we develop the task inclusion phase and conflict resolution phase to adapt to the resource constraint. The criteria of adding and removing tasks are modified, and the consensus rules are also revised. Third, a two-stage coalition formation method is proposed to form coalition for the tasks that cannot be performed by a single UAV, which makes full use of the remaining resources of UAVs. The coalition formation method first selects the UAVs which can provide the required resources and do not violate the deadline constraints as preliminary coalition members in the first stage. Then, the coalition size is further reduced by three criteria and the final coalition is obtained. Finally, a reassignment application of two-stage coalition formation method is also introduced to cope with the new task appearing in the dynamic environment. Simulation results illustrate the allocation process of RCPIA in detail when assigning known tasks and new tasks in the design scenario. In addition, to demonstrate the efficiency, RDAATSM is conducted to compare with RCPIA. The results indicate that RCPIA performs well in

resource-constrained task allocation problem and increases the robustness of assignment results.

The limitation of RCPIA is that it cannot balance the solution quality and computation time. In our future work, we plan to: 1) further improve the time efficiency of RCPIA; 2) study the coalition formation method with the precedence constraints; 3) study the impact of network topology on task assignment results.

Appendix

See Table 11.

Table 11 Action Rule For UAV i Based On Communication With UAV k Regarding Task k

UAV k (sender) thinks β_{kj} is	UAV i (receiver) thinks β_{kj} is	Receiver's Action (default:leave)
k	i	if $\eta_{kj} > \eta_{ij} \rightarrow$ update if $\eta_{kj} = \eta_{ij}$ and $\omega_{kj}^{\ominus} > \omega_{ij}^{\ominus} \rightarrow$ update
	k	update
	$m \notin \{i, k\}$	if $TS_{km} > TS_{im}$ or $\eta_{kj} > \eta_{ij} \rightarrow$ update if $\eta_{kj} = \eta_{ij}$ and $\omega_{kj}^{\ominus} > \omega_{ij}^{\ominus} \rightarrow$ update
	none	update
i	i	leave
	k	reset
	$m \notin \{i, k\}$	if $TS_{km} > TS_{im} \rightarrow$ reset
	none	leave
$m \notin \{i, k\}$	i	if $TS_{km} > TS_{im}$ if $\eta_{kj} > \eta_{ij} \rightarrow$ update elseif $\eta_{kj} = \eta_{ij}$ and $\omega_{kj}^{\ominus} > \omega_{ij}^{\ominus} \rightarrow$ update
	k	if $TS_{km} > TS_{im} \rightarrow$ update else $\rightarrow$ reset
	m	if $TS_{km} > TS_{im} \rightarrow$ update
	$n \notin \{i, k, m\}$	if $TS_{km} > TS_{im}$ and $TS_{kn} > TS_{in} \rightarrow$ update if $TS_{km} > TS_{im}$ if $\eta_{kj} > \eta_{ij} \rightarrow$ update elseif $\eta_{kj} = \eta_{ij}$ and $\omega_{kj}^{\ominus} > \omega_{ij}^{\ominus} \rightarrow$ update if $TS_{kn} > TS_{in}$ and $TS_{km} > TS_{im} \rightarrow$ reset
	none	if $TS_{km} > TS_{im} \rightarrow$ update
none	i	leave
	k	update
	$m \notin \{i, k\}$	if $TS_{km} > TS_{im} \rightarrow$ update
	none	leave

References

1. Alshawi MA, Shalan MB (2017) Minimal time dynamic task allocation for a swarm of robots. International Journal of Mechanical Engineering and Robotics Research 6(6)
2. Arif MU, Haider S (2018) A flexible evolutionary algorithm for task allocation in multi-robot team. In: International Conference on Computational Collective Intelligence, Springer, pp 89–99
3. Badreldin M, Hussein A, Khamis A (2013) A comparative study between optimization and market-based approaches to multi-robot task allocation. Advances in Artificial Intelligence (16877470)
4. Bayram H, Bozma HI (2016) Coalition formation games for dynamic multirobot tasks. The International Journal of Robotics Research 35(5), 514–527
5. Cao Y, Yu W, Ren W, Chen G (2012) An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial Informatics 9(1), 427–438
6. Chen J, Sun D (2011) Resource constrained multirobot task allocation based on leader-follower coalition methodology. The International Journal of Robotics Research 30(12), 1423–1434
7. Chen J, Sun D (2012) Coalition-based approach to task allocation of multiple robots with resource constraints. IEEE Transactions on Automation Science and Engineering 9(3), 516–528
8. Choi HL, Brunet L, How JP (2009) Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics 25(4), 912–926
9. Czarnecki E, Dutta A (2021) Scalable hedonic coalition formation for task allocation with heterogeneous robots. Intelligent Service Robotics 14(3), 501–517
10. Deng Q, Yu J, Wang N (2013) Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes. Chinese Journal of Aeronautics 26(5), 1238–1250
11. Farinelli A, Iocchi L, Nardi D (2017) Distributed on-line dynamic task assignment for multi-robot patrolling. Autonomous Robots 41(6), 1321–1345
12. Fu X, Wang H, Li B, Gao X (2018) An efficient sampling-based algorithms using active learning and manifold learning for multiple unmanned aerial vehicle task allocation under uncertainty. Sensors 18(8):2645
13. Fu X, Feng P, Gao X (2019a) Swarm uavs task and resource dynamic assignment algorithm based on task sequence mechanism. IEEE Access 7:41090–41100
14. Fu X, Zhang J, Zhang L, Chang S (2019b) Coalition formation among unmanned aerial vehicles for uncertain task allocation. Wireless Networks 25(1), 367–377
15. Han X, Bui H, Mandal S, Pattipati KR, Kleinman DL (2012) Optimization-based decision support software for a team-in-the-loop experiment: Asset package selection and planning. IEEE Transactions on Systems, Man, and Cybernetics: Systems 43(2), 237–251
16. Han X, Mandal S, Pattipati KR, Kleinman DL, Mishra M (2013) An optimization-based distributed planning algorithm: a blackboard-based collaborative framework. IEEE Transactions on Systems, Man, and Cybernetics: Systems 44(6), 673–686
17. Huang L, Qu H, Zuo L (2018) Multi-type uavs cooperative task allocation under resource constraints. IEEE Access 6:17841–17850
18. Ji X, Niu Y, Shen L (2016) Robust satisficing decision making for unmanned aerial vehicle complex missions under severe uncertainty. PloS one 11(11)
19. Jia Z, Yu J, Ai X, Xu X, Yang D (2018) Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm. Aerospace Science and Technology 76:112–125
20. Kan X, Thayer TC, Carpin S, Karydis K (2021) Task planning on stochastic aisle graphs for precision agriculture. IEEE Robotics and Automation Letters 6(2), 3287–3294
21. Kapoutsis AC, Chatzichristofis SA, Doitsidis L, de Sousa JB, Pinto J, Braga J, Kosmatopoulos EB (2016) Real-time adaptive multi-robot exploration with application to underwater map construction. Autonomous Robots 40(6), 987–1015
22. Khamis A, Hussein A, Elmogy A (2015) Multi-robot task allocation: A review of the state-of-the-art. In: Cooperative Robots and Sensor Networks 2015, Springer, pp 31–51
23. Kim MH, Baik H, Lee S (2015) Resource welfare based task allocation for uav team with resource constraints. Journal of Intelligent & Robotic Systems 77(3–4), 611–627
24. Korsah GA, Stentz A, Dias MB (2013) A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research 32(12), 1495–1512

25. Lim WH, Isa NAM (2015) Particle swarm optimization with dual-level task allocation. Engineering Applications of Artificial Intelligence 38:88–110
26. Muhuri PK, Rauniyar A (2017) Immigrants based adaptive genetic algorithms for task allocation in multi-robot systems. International Journal of Computational Intelligence and Applications 16(04):1750025
27. Nayak S, Yeotikar S, Carrillo E, Rudnick-Cohen E, Jaffar MKM, Patel R, Azarm S, Herrmann JW, Xu H, Otte M (2020) Experimental comparison of decentralized task allocation algorithms under imperfect communication. IEEE Robotics and Automation Letters 5(2), 572–579
28. Nedjah N, de Mendonça RM, de Macedo Mourelle L (2015) Pso-based distributed algorithm for dynamic task allocation in a robotic swarm. In: ICCS, pp 326–335
29. Nelke SA, Okamoto S, Zivan R (2020) Market clearing-based dynamic multi-agent task allocation. ACM Transactions on Intelligent Systems and Technology (TIST) 11(1):1–25
30. Nunes E, Manner M, Mitiche H, Gini M (2017) A taxonomy for task allocation problems with temporal and ordering constraints. Robotics and Autonomous Systems 90:55–70
31. Oh G, Kim Y, Ahn J, Choi HL (2017) Market-based distributed task assignment of multiple unmanned aerial vehicles for cooperative timing mission. Journal of Aircraft 54(6), 2298–2310
32. Torreño A, Onaindia E, Komenda A, Štolba M (2018) Cooperative multi-agent planning: A survey. ACM Computing Surveys (CSUR) 50(6):84
33. Turner J, Meng Q, Schaefer G, Whitbrook A, Soltoggio A (2017) Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system. IEEE Transactions on Cybernetics 48(9), 2583–2597
34. Wu H, Li H, Xiao R, Liu J (2018) Modeling and simulation of dynamic ant colony's labor division for task allocation of uav swarm. Physica A: Statistical Mechanics and its Applications 491:127–141
35. Wu X, Yin Y, Xu L, Wu X, Meng F, Zhen R (2021) Multi-uav task allocation based on improved genetic algorithm. IEEE Access 9:100369–100379
36. Xie S, Zhang A, Bi W, Tang Y (2019) Multi-uav mission allocation under constraint. Applied Sciences 9(11):2184
37. Xu Y, Sun Z, Xue X, Gu W, Peng B (2020) A hybrid algorithm based on mosfla and ga for multi-uavs plant protection task assignment and sequencing optimization. Applied Soft Computing 96:106623
38. Ye F, Chen J, Sun Q, Tian Y, Jiang T (2021) Decentralized task allocation for heterogeneous multi-uav system with task coupling constraints. The Journal of supercomputing 77(1):111–132
39. Zhai XB, Li L, Zhao X, Zhao Y, Liu K (2021) Real-time task allocation of heterogeneous unmanned aerial vehicles for search and prosecute mission. Wireless Communications and Mobile Computing 2021
40. Zhang A, Zhou D, Yang M, Yang P (2018) Finite-time formation control for unmanned aerial vehicle swarm system with time-delay and input saturation. IEEE Access 7:5853–5864
41. Zhang K, Collins Jr EG, Shi D (2012) Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 7(2):1–22
42. Zhang Y, Parker LE (2013) Considering inter-task resource constraints in task allocation. Autonomous Agents and Multi-Agent Systems 26(3), 389–419
43. Zhao W, Meng Q, Chung PW (2015) A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. IEEE Transactions on Cybernetics 46(4), 902–915
44. Zhen Z, Wen L, Wang B, Hu Z, Zhang D (2021) Improved contract network protocol algorithm based cooperative target allocation of heterogeneous uav swarm. Aerospace Science and Technology 119:107054
45. Zhou X, Wang H, Ding B, Hu T, Shang S (2019) Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. Expert Systems with Applications 116:10–20
46. Zitouni F, Harous S, Maamri R (2020) A distributed approach to the multi-robot task allocation problem using the consensus-based bundle algorithm and ant colony system. IEEE Access 8:27479–27494