drones

MDPI

Article

Distributed Task Allocation for a Multi-UAV System with Time Window Constraints

Wei Cui [1], Ruilin Li [1], Yanxiang Feng [1,2] and Yikang Yang [1,2,*]

[1] School of Automation Science and Engineering, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2] State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China
* Correspondence: yangyk74@mail.xjtu.edu.cn

Abstract: This paper develops a distributed method, namely, distributed allocation with time windows (DATW) for managing the multi-UAV task assignment problem (MTAP) with complex time window constraints. By aiming directly to minimize the average task completion time, the pro-posed DATW intends to achieve a conflict-free result that allocates all tasks within the validity time windows. Based on the decentralized PI (Performance Impact) framework, the proposed algorithm addresses the MTAP in a three-phase task assignment strategy, which includes task inclusion, conflict resolution, and task reallocation. The newly introduced task allocation phase achieves a noteworthy increase in an average number of allocated tasks. Unlike the traditional PI methods, the start time of each task is broadcasted among agents via communication typology, and the significance value of each task is directly related to its validity time window, such that the vast majority of tasks are able to be assigned properly without imposing any extra communication burdens. In the obtained conflict-free allocation solution by DATW, each task is allocated to a proper UAV with all given constraints satisfied. Finally, the simulation results demonstrate the effectiveness and superiority of the proposed DATW. Compared with existing (CBBA-based) solutions, results show up to an 18% increase in success rate (SR) using the proposed method.

Keywords: multi-UAV system; time window constraint; task allocation; distributed algorithm

Citation: Cui, W.; Li, R.; Feng, Y.; Yang, Y. Distributed Task Allocation for a Multi-UAV System with Time Window Constraints. *Drones* **2022**, *6*, 226. https://doi.org/10.3390/drones6090226

Academic Editor: Carlos Tavares Calafate

Received: 6 August 2022
Accepted: 28 August 2022
Published: 30 August 2022

1. Introduction

Currently, multiagent systems (e.g., unmanned aerial vehicle systems) have drawn increasing attention from researchers worldwide [1–6]. In comparison to a single agent, multiagent systems can achieve more complicated mission objectives and increase execution efficiency through cooperation. In this paper, we focus on the application of UAVs in multiagent systems due to the advantages of high flexibility, easy assembly, and low-cost consumption [7,8]. In particular, UAVs have been widely used in a variety of industries, including search and rescue, military reconnaissance, and traffic transportation. To date, there is a growing consensus that the multi-UAV task assignment problem (MTAP) has become one of the most challenging problems of multiagent systems.

In general, the MTAP is a combinatorial optimization problem with complex constraints that aims to coordinate a group of UAVs to accomplish various tasks while optimizing some given overall objectives [9]. Moreover, the MATP is proven to be a nondeterministic polynomial hard (NP-hard) problem [10], and as a result, the scale of the problem has a direct impact on the complexity of the solving process.

Early studies of the MATP focused on centralized approaches, where each UAV communicates situational awareness (SA) to a centralized server that generates a collaborative plan for the entire UAV swarm. Initially, the problem is modeled as mixed-integer linear programming (MILP) [11], which includes multiple processor resource allocation (MPRA) [12], the multiple traveling salesman problem (MTSP) [13], and the vehicle routing problem (VRP) [14]. By embedding complicated constraints into the models, the MILP algorithms

have advantages in describing the mission; however, the tree search method and exhaustive enumerations in MILP have resulted in an explosion in the computational demands for large-scale systems. Then, some heuristic algorithms [15–18], namely, swarm intelligence algorithms (SIAs), are proposed by searching a certain range of solution space in an acceptable time to obtain feasible solutions for MATP. Common heuristic methods include the genetic algorithm (GA) [15], ant colony algorithm [16], particle swarm optimization algorithm [17], wolf swarm algorithm [18], etc. Although these SIAs have a relatively straightforward structure, their stability is weak and a single point of failure can bring down the entire system [19].

Recently, some *distributed* allocation methods have been proposed, which are flexible and robust, consuming less communication and computational resources than centralized approaches. Most of these distributed approaches (e.g., contract network protocol) [20–23] are developed on the basis of market auction mechanisms. The UAVs place bids on tasks, and the highest bid wins the assignment. Choi et al., in [24], proposed the classical consensus-based bundle algorithm (CBBA) and introduced a *consensus procedure* to achieve agreement on the winning bid values through a consensus routine instead of SA. A series of modifications to the classical CBBA were made in [25,26] to expand the function and application of such a method. Then, another distributed approach called performance impact (PI) was proposed by Whitbrook et al. [27] and Zhao et al. [28], which aims to directly optimize the overall objective. Such methods introduce a key concept called significance to assess each task's contribution to the local time cost of an UAV. In comparison to CBBA, the PI method is developed on the basis of the heuristic optimization principle and can solve more time-sensitive MTAP while achieving allocation with lower time costs across tasks.

In realistic situations, the existence of various constraints and uncertainties will cause significant growth in the complexity of MATP [29]. To this end, most studies on MTAP describe this problem in terms of ideal mathematical models. For example, [30] proposes a "closed-loop CBBA" that considers the return of UAVs to the take-off base after the mission is completed, but ignores existing constraints during the allocation. Ponda et al., in [26], proposed an extension algorithm of CBBA to handle MTAP with time window constraints and vehicle fuel cost constraints; however, the allocated tasks in results notably decrease when time window constraints between tasks are more complex. In addition, PI-MaxAss was proposed as an extension of PI by Turner et al. [31], which aims to maximize the number of task allocations under strict time constraints. This article discusses task deadlines under a search and rescue scenario, while the time window constraints between tasks are ignored.

In this paper, common constraints such as time windows and UAV capabilities are combined to build a complex MTAP model. We propose an algorithm, namely, *distributed allocation with time windows* (DATW), which aims to achieve conflict-free allocation with a minimum average task completion time. The proposed DATW embeds time window constraints into the PI-based framework and iterates between three phases: the *task inclusion* phase, *conflict resolution* phase, and *task reallocation* phase. The first phase selects the optimal task within validity time windows for each UAV. The second reaches a consensus based on the significance value over all UAVs. The last allocates the tasks that remain unassigned after the former two phases due to time window constraints. Note that the time window constraints in this paper are "hard constraints", i.e., each task must start within its time window constraints; otherwise, the task is not allowed to be assigned. The key contributions of this paper are summarized as follows:

1. The time window constraints are first embedded into the PI-based framework. That is, the significance value of each task is directly related to its validity time window. In the obtained allocation, the vast majority of tasks are assigned by achieving a minimum average task completion time.
2. Compared with the PI method in [22], a new task reallocation phase that only considers the marginal significance value is proposed. The number of allocated tasks is

effectively improved by the aid of this third phase, where simulation results show that the average rate of feasible tasks has exceeded 70%.

3. The experiments reveal that the proposed DATW assigns more tasks within validity time windows than the existing E-CCBA method in [26]. In addition, the DATW has facilitated an 18% increase in success rate when assigning all tasks for UAVs.

This paper is organized as follows. Section 2 presents the description of MTAP and depicts the mathematical model of the distributed allocation with time window constraints. Section 3 proposes the PI-based distributed method DATW to solve the studied task allocation problem. Section 4 presents numerical simulations to demonstrate the merits of DATW. Finally, Section 5 concludes the paper.

2. Problem Description

This paper studies the multi-UAV task assignment problem (MTAP) with time window constraints. Let $N_t = \{t_1, t_2, \ldots, t_m\}$ be the set of m tasks and $N_u = \{u_1, u_2, \ldots, u_n\}$ be a set of n UAVs. A list of key symbols used hereafter is provided in Table 1. The properties of each $t_j \in N_t$ are denoted by a tuple $<loc(t_j), D_j>$, where $loc(t_j) = (x_j, y_j, z_j)$ is the coordinate of the task in the 3-D spatial plane and D_j is the duration required to perform task t_j. Meanwhile, an UAV $u_i \in N_u$ is denoted by $<loc(u_i), v_i>$, where $loc(u_i) = (x_i, y_i, z_i)$ is the initial position of u_i, and v_i is the flight velocity. UAVs communicate through a local communication topology denoted by a symmetric adjacency matrix G, where $G_{ih} = 1$ means u_i and u_h are capable of communicating with each other, while $G_{ih} = 0$ otherwise. We use $\boldsymbol{P}_i = <t_i^1, t_i^2, \ldots, t_i^{|Pi|}>$ to denote an ordered sequence of tasks assigned to u_i; UAV u_i starts from its initial coordinate $loc(u_i)$ and performs tasks in $\boldsymbol{P}_i$ sequentially. Once UAV u_i completes task t_i^j in $\boldsymbol{P}_i$, it immediately flies to the location of the next task t_i^{j+1}. The assigned task sequences for all UAVs constitute a *task allocation* $\boldsymbol{P} = [\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_n]^T$ of MTAP under consideration.

Table 1. Symbol Definitions.

Symbol	Definitions
N_t	Set of m tasks
N_u	Set of n UAVs
D_j	The duration required to perform task t_j
$loc(t_j)$	The spatial location of task t_j, $loc(t_j) = (x_j, y_j, z_j)$
$loc(u_i)$	The initial position of UAV u_i, $loc(u_i) = (x_i, y_i, z_i)$
v_i	The flight velocity of UAV u_i
G	The communication topology between UAVs
$\boldsymbol{P}_i$	An ordered sequence of tasks for UAV u_i
$\Phi(p, \boldsymbol{P}_i)$	The task which locates at the p-th position in the sequence $\boldsymbol{P}_i$
$T_j(\boldsymbol{P}_i)$	The time at which u_i starts to execute t_j
$\mathcal{F}(\boldsymbol{P}_i)$	The sum of the completion times for all tasks in $\boldsymbol{P}_i$, i.e., the local time cost of u_i
L_i	The execution capacity of u_i
$[T_{j\_start}, T_{j\_end}]$	The execution time window of task t_j
$\boldsymbol{Z}_i$	A task list stored on agent u_i that keeps track of which task is assigned to which UAV
$\boldsymbol{Q}_i$	A list stored on u_i that records the significance value for each task
$\boldsymbol{T}_i$	A list stored on u_i that records the start time for each task
$\boldsymbol{s}_i$	A list of timestamps for u_i
$\boldsymbol{P}_i \ominus t_j$	$\boldsymbol{P}_i$ with task t_j removed
$q_{ij}(\boldsymbol{P}_i \ominus t_j)$	The significance value of task t_j for u_i
$\boldsymbol{P}_i \oplus_k t_j$	$\boldsymbol{P}_i$ with task t_j inserted into the kth position
$q_{ij}^*(\boldsymbol{P}_i \oplus t_j)$	The marginal significance value of task t_j for u_i

On the basis of the above-mentioned information, the general working mechanism of the multi-UAV systems with time window is illustrated in Figure 1. Herein, the initial information board inputs all the relevant information of tasks and the underlying time

window constraints which should be satisfied by all UAVs. Specifically, for each UAV u_i, its reason system generates an allocation plan P_i consisted of tasks assigned to u_i and their ordered sequence. Whenever a u_i includes a new task, it updates the task's time information and transmits this information to another UAV u_j with $G_{ij} = 1$ via communication link. This reasoning system continuously updates the allocation plan according to newly received information. Based on such a dynamic reasoning system, all UAVs work in parallel and interact with each other. Thus, one key problem of our paper is to design an effective reasoning system to achieve a conflict-free allocation by satisfying all time window constraints.

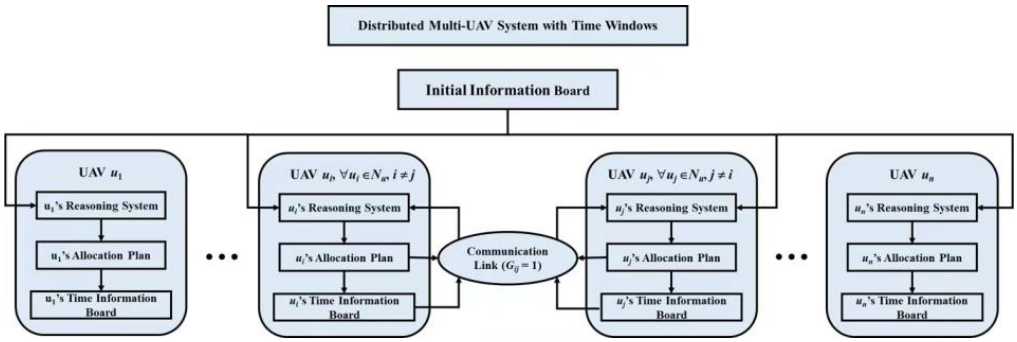

Figure 1. General system architecture for distributed multi-UAV systems with time windows.

For the sake of convenience, we use the notation $\Phi(\boldsymbol{P}_i, g) = t_j$ to denote the fact that task t_j is the g-th ($g > 1$) task in $\boldsymbol{P}_i$, i.e., $t_j = t_i^g$. If task t_k is t_j's preorder task, $\Phi(\boldsymbol{P}_i, g-1) = t_k$. Then, the time parameter $T_j(\boldsymbol{P}_i)$, when u_i starts to execute t_j, is calculated as follows:

$$T_j(\boldsymbol{P}_i) = \begin{cases} T_k(\boldsymbol{P}_i) + D_k + \frac{dis\left(loc(t_k), loc\left(t_j\right)\right)}{v_i}, & g > 1 \\ \frac{dis\left(loc(u_i), loc\left(\ t_j\right)\right)}{v_i}, & g = 1 \end{cases} \tag{1}$$

where $dis(\bullet)$ represents the distance between two different locations. Moreover, the completion time $F_j(\boldsymbol{P}_i)$ for task t_j is calculated as follows:

$$F_j(\boldsymbol{P}_i) = T_j(\boldsymbol{P}_i) + D_j \tag{2}$$

In the rest of this paper, let $\mathcal{F}(\boldsymbol{P}_i) = \sum_{t_k \in \boldsymbol{P}_i} F_k(\boldsymbol{P}_i)$ be the sum of the completion time for all tasks in $\boldsymbol{P}_i$; herein, $\mathcal{F}(\boldsymbol{P}_i)$ is called the *local time cost* of UAV u_i.

The practical MTAP is subject to the constraints related to UAVs and tasks:

1. *Conflict-freeness:* each task is only assigned to one UAV.
2. *Capability constraints:* each UAV can execute at most L_i tasks at a time. That is,

$$|\boldsymbol{P}_i| \leq L_i \ (1 \leq i \leq n) \tag{3}$$

3. *Time window constraints:* each task must be started during the interval of a time window. Let T_{j_start} be the earliest start time of task t_j, and T_{j_end} be the latest start time so that the validity interval for t_j is given as

$$T_j(\boldsymbol{P}_i) \in \left[T_{j\_start}\ , T_{j\_end}\right] \tag{4}$$

4. *Power consumption constraints:* the remaining fuel mass Fr_i for each UAV must be more than a threshold Δ. Let Fo_i be the initial fuel mass of UAV u_i, and u_i consumes the fuel at a nominal rate vm_i, such that the power consumption constraint is given as follows

$$Fr_i = Fo_i - vm_i \times \mathcal{F}(P_i) \ \geq \ \Delta \tag{5}$$

This paper aims to find the optimal task allocation for MATP with the goals of satisfying the above-mentioned constraints and minimizing the *global time cost*, which is the sum of all local time costs across all UAVs. The sought MTAP can be modeled mathematically as follows:

$$J = min\left\{\sum_{i=1}^{n} \mathcal{F}(\boldsymbol{P}_i)\right\} \tag{6}$$

$$s.t. \qquad \bigcup_{i=1}^{n} \boldsymbol{P}_i = N_t, \boldsymbol{P}_i \bigcap \boldsymbol{P}_j = \varnothing, i \neq j \tag{7}$$

$$|\boldsymbol{P}_i| \leq L_i, \quad \forall u_i \in N_u \tag{8}$$

$$T_{j\_start} \leq T_j(\boldsymbol{P}_i) \leq T_{j\_end}, \ \forall t_j \in \boldsymbol{P}_i, \forall u_i \in N_u \tag{9}$$

$$Fr_i = Fo_i - vm_i \times \mathcal{F}(P_i) \ \geq \ \Delta, \ \forall u_i \in N_u \tag{10}$$

where objective (6) minimizes the global time cost; (7) ensures that the allocation includes all tasks and each task is assigned to only one UAV, indicating conflict-freeness; (8) ensures that the number of tasks assigned to each UAV u_i does not exceed the execution capacity L_i; (9) guarantees the satisfaction of the time window constraint for all tasks; (10) ensures that each UAV satisfies the power consumption constraints.

3. Distributed Allocation with Time Window Constraints

This section proposes a heuristic method, namely, *distributed allocation with time windows* (DATW), for solving the MATP under consideration. The DATW algorithm is run concurrently on all UAVs that transmit information to each other via local communication. The proposed DATW algorithm consists of three phases: *task inclusion*, *conflict resolution*, and *task reallocation*. The first two phases are performed iteratively until an initial allocation is agreed upon by all UAVs, although some tasks may remain unassigned. Then, the last phase aims to maximize the allocation of these unassigned tasks to UAVs under the time window constraints.

Before introducing the DATW algorithm, some message or information stored on each UAV u_i is defined as follows.

- *Task list:* $\boldsymbol{Z}_i = [Z_{i1}, \ldots, Z_{im}]^T$ is a vector that keeps track of which task is assigned to which UAV. The entry $Z_{ij} = k$ if u_i thinks that task t_j is assigned to u_k, while Z_{ij} is infinity, i.e., $Z_{ij} \to \infty$, if u_i deems t_j to be unassigned;
- *Task significance list:* $\boldsymbol{Q}_i = [Q_{i1}, \ldots, Q_{im}]^T$ is a vector recording the significance values of all tasks; the concept of the significance of tasks will be given in Section 3.1. If $Z_{ij} \to \infty$, we set $Q_{ij} \to \infty$;
- *Task start time list:* $\boldsymbol{T}_i = [T_{i1}, \ldots, T_{im}]^T$ is a vector recording the begin-execute time for each task. When $Z_{ij} = k$, T_{ij} reflects the time when UAV u_k starts to execute task t_j. Note that if $Z_{ij} \to \infty$, $T_{ij} \to \infty$;
- *Timestamp list*: $\boldsymbol{s}_i = [s_{i1}, \ldots, s_{in}]^T$ records the timestamp when u_i obtains the latest message from each of the other UAVs. Once a message is passed, the timestamp s_{ih} is calculated as follows:

$$s_{ih} = \begin{cases} \tau_r, & G_{ih} = 1 \\ max_{\alpha:G_{i\alpha}=1} s_{\alpha h} - 1, & G_{ih} \neq 1 \end{cases} \tag{11}$$

where τ_r represents the time when u_i receives the information.

3.1. Task Inclusion Phase

In the first phase, each UAV u_i selects the optimal task that can minimize u_i's local time cost and inserts this task into a proper position of list $\boldsymbol{P}_i$ so that the time window constraints are satisfied. The concepts of "significance" and "marginal significance" are given to describe how tasks affect the local time cost of an UAV.

1. *Significance*

Consider a task $t_j \in \boldsymbol{P}_i$; let $\boldsymbol{P}_i \ominus t_j$ represents the remaining task sequence after removing t_j from $\boldsymbol{P}_i$. Then, the significance value of t_j with regard to u_i is formulated as follows:

$$q_{ij}(\boldsymbol{P}_i \ominus t_j) = \left[\mathcal{F}(\boldsymbol{P}_i) - \mathcal{F}(\boldsymbol{P}_i \ominus t_j)\right] \times \left[T_j(\boldsymbol{P}_i) - T_{j\_start}\right] \tag{12}$$

By definition, $q_{ij}(\boldsymbol{P}_i \ominus t_j)$ is composed of two parts: $\mathcal{F}(\boldsymbol{P}_i) - \mathcal{F}(\boldsymbol{P}_i \ominus t_j)$ and $T_j(\boldsymbol{P}_i) - T_{j\_start}$. The first part equals the decrease in u_i's local time cost by removing t_j from $\boldsymbol{P}_i$, directly reflecting the contribution of t_j to the local time cost of u_i. Meanwhile, the second is the interval between the earliest start time allowed for t_j to start under the time window constraints and the time u_i starts to execute t_j. That is, agent u_i gives preference to tasks that are performed immediately upon meeting the time window constraints. In this case, both $T_j(\boldsymbol{P}_i)$ and $F_j(\boldsymbol{P}_i)$ for task t_j are further decreased, and thus, the contribution of t_j to the local time cost of u_i is logically decreased. Note that if $t_j \notin \boldsymbol{P}_i$, let $q_{ij}(\boldsymbol{P}_i \ominus t_j) \to \infty$.

2. *Marginal significance*

Considering a task $t_j \notin \boldsymbol{P}_i$, the local time cost of UAV u_i is increased after inserting t_j into $\boldsymbol{P}_i$. Then, the marginal significance, denoted by $q_{ij}^*(\boldsymbol{P}_i \oplus t_j)$, is required to reflect the minimum increase in u_i's local time cost by adding t_j into $\boldsymbol{P}_i$, which is formulated as follows:

$$q_{ij}^*(\boldsymbol{P}_i \oplus t_j) = min_{k=1}^{|\boldsymbol{P}_i|+1}\left[\mathcal{F}(\boldsymbol{P}_i \oplus_k t_j) - \mathcal{F}(\boldsymbol{P}_i)\right] \times \left[T_j(\boldsymbol{P}_i) - T_{j\_start}\right] \tag{13}$$

where $\boldsymbol{P}_i \oplus_k t_j$ represents the task sequence after inserting t_j into the kth position of $\boldsymbol{P}_i$. Note that $q_{ij}^*(\boldsymbol{P}_i \oplus t_j) \to \infty$ if $t_j \in \boldsymbol{P}_i$. Moreover, if $t_j \in \boldsymbol{P}_i$, $Z_{ij} = i$ and $Q_{ij} = q_{ij}(\boldsymbol{P}_i \ominus t_j)$.

In this task inclusion phase, a task t_j is allowed to be assigned to the UAV u_i if the following conditions hold:

(a). $Z_{ij} \neq i$ and $q_{ij}^*(\boldsymbol{P}_i \oplus t_j) < Q_{ij}$;

(b). Let $p = \arg q_{ij}^*(\boldsymbol{P}_i \oplus t_j)$; after inserting t_j into the p-th position of $\boldsymbol{P}_i$, all tasks in the sequence $\boldsymbol{P}_i \oplus_p t_j$ satisfy the underlying time window constraints;

(c). $Fo_i - vm_i \times \mathcal{F}(\boldsymbol{P}_i \oplus t_j) \geq \Delta$.

The above condition (a) ensures that t_j is unassigned to u_i and requires that the marginal significance value $q_{ij}^*(\boldsymbol{P}_i \oplus t_j)$ for t_j is strictly less than its significance Q_{ij} stored on u_i, such that the global time cost might be decreased by assigning t_j to u_i. Condition (b) guarantees the satisfaction of time window constraints after inserting t_j into $\boldsymbol{P}_i$, while condition (c) ensures the satisfaction of power consumption constraints.

Let $\wp(\boldsymbol{P}_i) = \{t_j \in N_t \mid \text{Conditions (a), (b), and (c) both hold for } t_j \text{ and } u_i\}$ be the set of tasks that satisfy the above conditions. This phase chooses the task $t_k \in \wp(\boldsymbol{P}_i)$ expressed in (14) and inserts it into the position $p = \arg q_{ik}^*(\boldsymbol{P}_i \oplus t_k)$ of list $\boldsymbol{P}_i$.

$$t_k = argmax_{t_j \in \wp(\boldsymbol{P}_i)}(Q_{ij} - q_{ij}^*(\boldsymbol{P}_i \oplus t_j)) \tag{14}$$

The corresponding information related to task t_k is updated as $Z_{ik} = i$, $Q_{ik} = q_{ik}^*(\boldsymbol{P}_i \oplus t_k)$, and the start time for all tasks in $\boldsymbol{P}_i$ is recalculated.

The above task addition process is repeated until $\wp(\boldsymbol{P}_i) = \emptyset$ or $|\boldsymbol{P}_i| = L_t$, which indicates that no task can be included in $\boldsymbol{P}_i$. Since the inclusion of new tasks may change the significance of other existing tasks in $\boldsymbol{P}_i$, the list $\boldsymbol{Q}_i$ is required to update accordingly (12) after inclusion. The whole procedure is depicted in Algorithm 1.

Algorithm 1: u_i execution task adding process at iteration λ.

Initialization: $\boldsymbol{P}_i(\lambda) = \boldsymbol{P}_i(\lambda - 1)$, $Z_i(\lambda) = Z_i(\lambda - 1)$, $Q_i(\lambda) = Q_i(\lambda - 1)$, $T_i(\lambda) = T_i(\lambda - 1)$.
1: **while** $|P_i| \leq L_i$ **do**
2: For each task $t_j \in \boldsymbol{P}_i$ compute the marginal significance value $q_{ij}{}^*(\boldsymbol{P}_i \oplus t_j)$;
3: Compute $\wp(\boldsymbol{P}_i) \leftarrow \{t_j \in N_t \mid$ Conditions (a), (b) and (c) both hold for t_j and $u_i\}$;
4: **if** $\wp(\boldsymbol{P}_i) \neq \emptyset$ **then**
5: $t_k \leftarrow argmax_{t_j \in \wp(\boldsymbol{P}_i)}(Q_{ij} - q_{ij}{}^*(\boldsymbol{P}_i \oplus t_j))$;
6: $p \leftarrow \arg q_{ij}{}^*(\boldsymbol{P}_i \oplus t_k)$;
7: Insert task t_k into $p -$ th of $\boldsymbol{P}_i$;
8: $Z_{ik} \leftarrow i$;
9: $Q_{ik} \leftarrow q_{ij}{}^*(\boldsymbol{P}_i \oplus t_j)$;
10: Update $\boldsymbol{T}_i$;
11: **end if**
12: **else**
13: **break**;
14: **end if**
15: **end while**
16: Update $\boldsymbol{Q}_i$;

3.2. Conflict Resolution Phase

In the previous task inclusion phase, a task may be assigned to more than one UAV, as Algorithm 1 runs concurrently on all UAVs, leading to a conflict. Thus, the second phase is conducted to obtain a conflict-free allocation via local communication topology. In particular, the conflict resolution phase is composed of two stages: (i) *consensus*, where UAVs communicate with each other for consensus, and (ii) *task removal*, where UAVs determine whether to remove a task from its current task list. These two stages are repeated alternately on each UAV until the global consensus is achieved across all UAVs, i.e., $\boldsymbol{Q}_i = \boldsymbol{Q}_h$, $\forall u_i, u_h \in N_u$, and $u_i \neq u_h$.

3.2.1. Consensus Stage

To search for the smallest significance value for every task, in this stage, each UAV u_i transmits the relevant information, including Z_i, Q_i, T_i, and s_i, to its neighbor UAV u_j through local communications.

Suppose that UAV u_i receives a message from u_h where $G_{ih} = 1$. It determines whether the received message is the latest for task t_j based on lists $\boldsymbol{Z}_i$ and $\boldsymbol{s}_i$, and then, three possible actions are taken on task t_j:

- Update: $Z_{ij} = Z_{hj}$, $Q_{ij} = Q_{hj}$, $T_{ij} = T_{hj}$;
- leave: $Z_{ij} = Z_{ij}$, $Q_{ij} = Q_{ij}$, $T_{ij} = T_{hj}$;
- reset: $Z_{ij} \rightarrow \infty$, $Q_{ij} \rightarrow \infty$, $T_{ij} \rightarrow \infty$.

Specifically, agents select actions according to the decision rules given in Table 2. The first two columns of Table 2 record the executor of task t_j believed by sender u_h and receiver u_i, respectively. The third column outlines the action taken on t_j with respect to receiver u_i. Note that once a piece of message is passed, timestamp g_i is updated according to Formula (11) to obtain the latest time information.

3.2.2. Task Removal Stage

After the consensus stage, each UAV u_i checks the current task sequence $\boldsymbol{P}_i$ and removes tasks in two sets $A = \{t_j \in \boldsymbol{P}_i \mid Z_{ij} \neq i\}$ and $B = \{t_j \in \boldsymbol{P}_i \mid T_j(\boldsymbol{P}_i) < T_{j\_start}$ or $T_j(\boldsymbol{P}_i) > T_{j\_end}\}$. Herein, A denotes the set of tasks in $\boldsymbol{P}_i$ whose corresponding information Z_{ij} has been changed after the communication, while B represents the set of tasks in $\boldsymbol{P}_i$ violating the time window constraints.

Table 2. UAVs communication rule.

Z_{hj} (u_h is Sender)	Z_{ij} (u_i is Receiver)	u_i's Action
h	i	if $Q_{hj} < Q_{ij}$: update
	h	update
	$\alpha \notin \{h,i\}$	if $s_{h\alpha} > s_{i\alpha}$ or $Q_{hj} < Q_{ij}$: update
	∞	update
i	i	leave
	h	reset
	$\alpha \notin \{h,i\}$	if $s_{h\alpha} > s_{i\alpha}$: reset
	∞	leave
$\alpha \notin \{h,i\}$	i	if $s_{h\alpha} > s_{i\alpha}$ and $Q_{hj} < Q_{ij}$: update
	h	if $s_{h\alpha} > s_{i\alpha}$: update
		other case: reset
	α	if $s_{h\alpha} > s_{i\alpha}$: update
	$\beta \notin \{h,I,m\}$	if $s_{h\alpha} > s_{i\alpha}$ and $s_{h\beta} > s_{i\beta}$: update
		if $s_{h\alpha} > s_{i\alpha}$ and $Q_{hj} < Q_{ij}$: update
		if $s_{h\beta} > s_{i\beta}$ and $s_{i\alpha} > s_{h\alpha}$: reset
	∞	if $s_{h\alpha} > s_{i\alpha}$: update
∞	i	leave
	h	update
	$\alpha \notin \{h,i\}$	if $s_{h\alpha} > s_{i\alpha}$: update

First, a task in A can be removed from $\boldsymbol{P}_i$ and A, according to the following criterion:

$$max_{t_j \in A} \left(\overline{q_{ij}} - Q_{ij} \right) > 0 \tag{15}$$

where $\overline{q_{ij}} = \mathcal{F}(\boldsymbol{P}_i) - \mathcal{F}(\boldsymbol{P}_i \ominus t_j)$ represents the significance of t_j computed from the current task sequence $\boldsymbol{P}_i$, while Q_{ij} denotes t_j's significance obtained from the consensus stage.

We release a task t_k from $\boldsymbol{P}_i$ and A such that $t_k = argmax_{t_j \in A} \left(\overline{q_{ij}} - Q_{ij} \right)$. This removal process is continued until (15) is not satisfied or A is empty. When this process is terminated but $A \neq \emptyset$, each remaining task $t_k \in A$ should be returned to agent u_i by setting $Z_{ik} = u_i$.

Next, each UAV u_i checks its sequence $\boldsymbol{P}_i$ and releases the tasks in B sequentially. Specifically, once there is a task $t_j \in \boldsymbol{P}_i$ that does not meet the time window constraint, t_j is removed from the sequence $\boldsymbol{P}_i$ immediately, and the related information is updated as $Z_{ij} \to \infty$, $Q_{ij} \to \infty$, $T_{ij} \to \infty$. Note that each time u_i removes a task from $\boldsymbol{P}_i$, the start time for other tasks in $\boldsymbol{P}_i$ is updated. Thus, we need to update set B after each removal. Such a removal process is repeated alternately until all tasks in all $\boldsymbol{P}_i$ satisfy the time window constraint. At the end of the task removal stage, the relevant information Q_{ij} and T_{ij} for each $t_j \in \boldsymbol{P}_i$ is updated correspondingly.

In the conflict resolution phase, the algorithm iterates over the consensus stage and task removal stage until a global consensus is achieved across all UAVs. To this end, we obtain a conflict-free allocation where tasks are assigned by meeting the time window constraints. The whole procedure is illustrated by Algorithm 2.

Algorithm 2: Conflict resolution phase of UAV u_i at iteration λ.

Initialization: $\boldsymbol{P}_i(\lambda) = \boldsymbol{P}_i(\lambda-1)$, $\boldsymbol{Z}_i(\lambda) = \boldsymbol{Z}_i(\lambda-1)$, $\boldsymbol{Q}_i(\lambda) = \boldsymbol{Q}_i(\lambda-1)$, $\boldsymbol{T}_i(\lambda) = \boldsymbol{T}_i(\lambda-1)$, set $\boldsymbol{s}_i$ to zero vector.

Send $\boldsymbol{Q}_i$, $\boldsymbol{T}_i$,$\boldsymbol{Z}_i$ and $\boldsymbol{s}_i$ to u_h where G_{ih}=1;
Receive $\boldsymbol{Q}_h$,$\boldsymbol{T}_h$,$\boldsymbol{Z}_h$ and $\boldsymbol{s}_h$ from u_h where G_{hi}=1;
Update $\boldsymbol{Q}_i$, $\boldsymbol{T}_i$,$\boldsymbol{Z}_i$ and $\boldsymbol{s}_i$ according to the rules in Table 2;
$A \leftarrow \{t_j \in \boldsymbol{P}_i \mid Z_{ij} \neq i, t_j \in \boldsymbol{P}_i\}$;
while $max_{t_j \in A}\left(\overline{q_{ij}} - Q_{ij}\right) > 0$ **do**
    $t_k \leftarrow argmax_{t_j \in A}\left(\overline{q_{ij}} - Q_{ij}\right)$;
    Remove task t_k from $\boldsymbol{P}_i$ and A;
end while
if $A \neq \emptyset$ **then**
    let $Z_{ij} = i$, $\forall t_j \in A$;
end if
$B \leftarrow \{t_j \in \boldsymbol{P}_i \mid T_j(\boldsymbol{P}_i) < T_{j\_start}$ or $T_j(\boldsymbol{P}_i) > T_{j\_end}\}$;
while $B \neq \emptyset$ **do**
    Remove task $t_k \in B$ from $\boldsymbol{P}_i$;
    Set Z_{ij}, Q_{ij}, T_{ij} as infinity values
    Update B;
end while

3.3. Task Reallocation Phase

After completing iterations between the above phases, the obtained allocation generally covers all tasks. However, in some cases, a few tasks may remain unassigned due to the strict time window constraints. Let Ca_i be the set of these unassigned tasks for each UAV u_i, i.e., $Ca_i = \{t_k \mid Z_{ik} \rightarrow \infty\}$. This phase intends to reassign tasks in Ca_i to some UAVs to increase the number of allocated tasks. That is, for each UAV u_i, only tasks in Ca_i can be added to or removed from the current list $\boldsymbol{P}_i$, while assigned task $t_j \notin Ca_i$ is not allowed to be moved. This phase is composed of two stages: *secondary inclusion* and *secondary conflict resolution*.

3.3.1. Secondary Inclusion

We define a vector $M_i = [m_{i1}, m_{i2}, \ldots, m_{i\,|Cai|}]^{\mathrm{T}}$ for all tasks in Ca_i with regards to UAV u_i, where the entry $m_{ij} = 1$ if a task $t_j \in Ca_i$ has been added to $\boldsymbol{P}_i$ but removed from the same sequence for violating time window constraint, and $m_{ij} = 0$ otherwise. By definition, a task $t_j \in Ca_i$ can be assigned to UAV u_i according to the following conditions:

(d). $Z_{ij} \neq i$ and $m_{ij} = 0$;

(e). let $p = \arg q_{ij}^*(\boldsymbol{P}_i \oplus t_j)$; after inserting t_j into the p-th position of $\boldsymbol{P}_i$, all tasks in the sequence $\boldsymbol{P}_i \oplus_p t_j$ satisfy the underlying time window constraint.

(f). $Fo_i - vm_i \times \mathcal{F}(\boldsymbol{P}_i \oplus t_j) \geq \Delta$.

Condition (d) ensures that tasks are not repeatedly added or removed by the same UAV, condition (e) ensures that all tasks in $\boldsymbol{P}_i \oplus t_j$ satisfy the time window constraints after inserting t_j into $\boldsymbol{P}_i$, and condition (f) ensures the satisfaction of power consumption constraints.

Let $\Re(\boldsymbol{P}_i, Ca_i) = \{t_j \in Ca_i \mid$ Conditions (d), (e) and (f) both hold for t_j and $u_i\}$ denote the set of tasks that satisfies the above conditions. To minimize the significance value, each UAV u_i chooses the task $t_k \in \Re(\boldsymbol{P}_i, Ca_i)$, according to (16), and inserts it into the position $p = \arg q_{ik}^*(\boldsymbol{P}_i \oplus t_k)$ of list $\boldsymbol{P}_i$.

$$t_k = argmin_{t_j \in \Re(\boldsymbol{P}_i,\, Ca_i)} q_{ij}^*(\boldsymbol{P}_i \oplus t_j) \tag{16}$$

Different from criterion (14), (16) requires that the UAV only selects out the unassigned task t_k with the minimum marginal significance value. The above process is repeated until no task can be included in $\boldsymbol{P}_i$, i.e., $\Re(\boldsymbol{P}_i, Ca_i) = \emptyset \mid \boldsymbol{P}_i \mid = L_i$.

3.3.2. Secondary Conflict Resolution

To obtain a conflict-free allocation, the secondary conflict resolution stage iterates between the consensus process and the task removal process until global consensus is achieved across all UAVs.

In the consensus process, the heuristic consensus rule proposed in Section 3.2.1 is used to obtain a conflict-free allocation. Note that each UAV only exchanges the information related to tasks in Ca_i with its neighboring UAVs through local communications. According to the decision rules given in Table 2, once u_i receives a message, it *updates*, *resets*, or *leaves* the stored information Z_{ij}, Q_{ij}, T_{ij}, and s_{ij} for each task $t_j \in Ca_i$.

Then, in the task removal process, each UAV u_i checks its current task sequence $\boldsymbol{P}_i$ and removes tasks belonging to two sets A'={$t_j \in \boldsymbol{P}_i \mid Z_{ij} \neq i$} and B'={$t_j \in \boldsymbol{P}_i \mid T_j(\boldsymbol{P}_i) < T_{j\_start}$ or $T_j(\boldsymbol{P}_i) > T_{j\_end}$} from $\boldsymbol{P}_i$. Herein, A' denotes the set of tasks whose information Z_{ij} is changed after the consensus process, while B' represents the set of tasks violating the time window constraints. The tasks in A' and B' are sequentially removed from $\boldsymbol{P}_i$ using the same method as in Section 3.2.2. In addition, each time a task $t_j \in B'$ is removed, we update its related information as $Z_{ij} \to \infty$, $Q_{ij} \to \infty$, $T_{ij} \to \infty$, and we also set $m_{ij} = 1$ such that task t_j is prevented from repeatedly assigning to $\boldsymbol{P}_i$, according to condition c). These requirements not only avoid invalid task inclusion and removal, but also improve the performance of the entire system.

The secondary inclusion and secondary conflict resolution are repeated alternately until no actions can be made for a period of time. That is, the task reallocation phase has already converged, and an optimized global conflict-free task assignment that satisfies all time window constraints is obtained.

3.4. Convergence Analysis

The first two phases of the DATW algorithm alternate until an initial allocation is obtained, and then, the last phase is performed to reallocate these unassigned tasks. In the ideal case, the system deems to be converged when no changes can be made in the first two phases. However, due to the strict time window constraints, some tasks may be repeatedly included and removed by the same UAV, and thus, an infinite cycle arises. To avoid such reiteration and reallocate those tasks in the last phase, iterations between the first two phases are terminated, provided that the allocations obtained after the second phase remain the same for a specified period.

Moreover, each UAV always aims to decrease the global time cost at each iteration. Specifically, the significance value of a task is highly related to its current contribution to the local time cost of its assigned UAV. In the first two phases, each UAV u_i exchanges the significance of all tasks with its neighbor UAVs, and then, based on these significance values, u_i tries to decrease the global time cost $\sum_{n=1}^{n} \mathcal{F}(\boldsymbol{P}i)$ by recursively adding or removing tasks from its task sequence. The last phase reassigns tasks in $Ca_{i,}$ which collects all unassigned tasks after the first two phases, to agents. In particular, each u_i selects unassigned tasks in Ca_i with the minimum significance value in each iteration of the last phase. Note that the initial significance value for each task is set to infinity and is continuously updated as the significance of a task is highly correlated with the current allocation. Consequently, the DATW algorithm converges when the task significance list of each UAV is not changed for a specified period of time.

Formally, the proposed DATW running on each UAV can be expressed in detail as follows.

Step 1: Performing the task inclusion phase, according to Algorithm 1.

Step 2: Running Algorithm 2. The conflict resolution phase is repeated until global consensus is achieved over all UAVs.

Step 3: Step 1 and Step 2 are repeated until the allocation obtained after Step 2 remains the same for a specified period of time.

Step 4: Carrying out the secondary inclusion stage to assign tasks in Ca_i to agents.

Step 5: Carrying out the secondary conflict resolution stage, performing the consensus process and the task removal process alternately, until the global consensus is achieved over all UAVs.

Step 6: Step 4 and Step 5 are repeated until no changes have been made in the task significance list Q_i for a specified period of time, and then, the algorithm is completed.

Moreover, we obtain the working architecture of the DATW by integrating the above 6steps through the process in Figure 2.

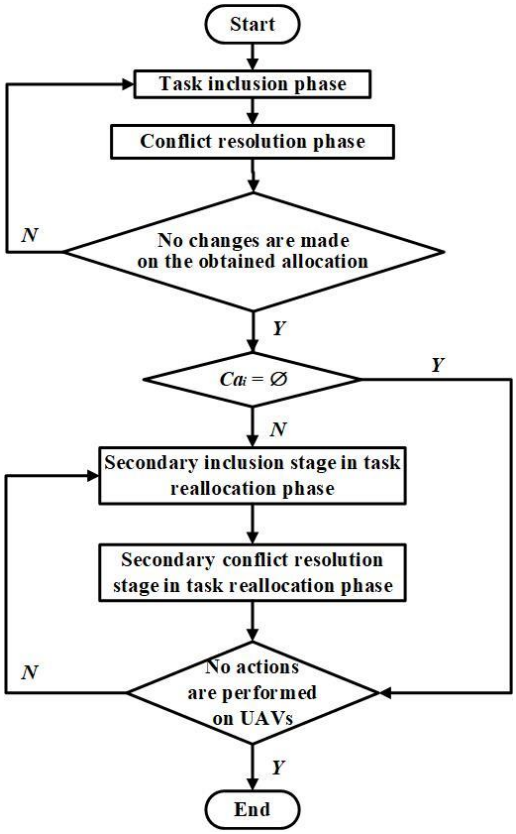

Figure 2. Working architecture for the distributed allocation with time windows.

4. Simulation

In this section, numerical simulations are conducted to assess the effectiveness of the proposed DATW algorithm. The proposed method is first verified across search and rescue scenarios with different settings generated randomly. Then, the proposed method is compared with an extension of the CBBA (E-CBBA) method in [26] in terms of various parameter settings.

4.1. Scenario and Simulation Setup

The performance of the DATW method is evaluated based on the *search and rescue* scenarios proposed in [28]. Suppose that a number of survivors are discovered in a specific area, and a group of UAVs are required to perform rescue tasks on these survivors. The survivors are classified as A-type, B-type, and C-type according to their health condition. All survivors are considered to have equal priority but must be rescued within the different time window constraints summarized in Table 3. It is assumed that one-quarter of the survivors are A-type, one-quarter are B-type and the remaining ones are C-type. The goal is to minimize the average time cost across all rescue tasks and to rescue as many survivors as possible upon the validity time interval. Moreover, in such a scenario, all UAVs and survivors are randomly initialized in a 10,000 × 10,000 × 1000 m 3-D spatial plane. The UAVs' speeds are assumed to always be 50 m/s, and each UAV takes 30 s to rescue a

survivor but executes at most five tasks at a time. Moreover, the UAVs' initial fuel masses are set as 400kg, and each UAV consumes the fuel at 0.05kg/s.

Table 3. Time Window Constraints [T_{j_strart}, T_{j_end}] of rescue missions for A, B, and C survivors.

Survivors' Type	Range of Values for the Earliest Start Time T_{j_strart} of a Rescue Mission	Range of Values for the Latest Start Time T_{j_end} of Rescue Missions
A	[0 s, 20 s]	[100 s, 120 s]
B	[50 s, 200 s]	[300 s, 400 s]

To better describe the problem size of various scenarios, we introduce a new parameter, namely, the task-to-UAV ratio (TUR), represented by $\delta \equiv m/n$, where m (resp. n) is the number of tasks (resp. UAVs). Table 4 shows the parameters of all testing scenarios where the UAV number is denoted by $n \in \{3, 4, 9, 10, 15, 16\}$ and the corresponding TUR is set as $\delta \in \{1, 2, 3\}$. Hence, there are $18 = 6 \times 3$ combinations of scenarios to consider. The time windows for all these combinations are randomly generated, according to Table 3.

Table 4. Parameter Size for Each Instance Type.

Instance Type	UAV Num	TUR
Small	3	$\delta = 1, 2, 3$
	4	
Medium	9	
	10	
Large	15	
	16	

4.2. Simulation Steps

In this paper, all experiments are performed on an Intel(R) Core (TM) i9-10900CPU@3.70 GHz PC, and the simulation program is executed with Python under the Windows 10 operating system. Four basic libraries in Python are employed to simulate the entire process: NumPy, Pandas, Matplotlib, and Time. Specifically, the NumPy is used to handle the calculations of a matrix; the Pandas is used to export the generated data in the simulation and perform further analysis; the Matplotlib is used for drawing two-dimensional diagrams, while the Time is added for addressing problems with time information.

The simulation program consists of a clock module, a data generation module, a task assignment module, and an output module. The clock module, which uses functions from the Time library, is the first component of the simulation. The clock module keeps track of the start and end time of each simulation, which aids in evaluating the computational complexity of the whole algorithm. Meanwhile, the clock module also records the timestamp $\boldsymbol{s_i}$ when each UAV u_i obtains the latest message from the other UAVs. Then, the data generation module is introduced and is based on the functions in the NumPy library. In particular, this module randomly generates the initial scenario for each simulation, according to the settings in Section 4.1. To this end, the task assignment module imports the above-mentioned data and obtains the proper allocation for each UAV. Ultimately, the output module is employed based on the functions in the Pandas and the Matplotlib, such that the visual output of the allocation results is achieved.

4.3. Simulation Results

4.3.1. Validation Experiments

A series of validation experiments are conducted to evaluate the performance of DATW; there are four performance parameters to consider:

- G: the average completion time of all tasks, which is the objective of the studied MTPA and can be obtained from (6);
- Λ: the number of iterations, which is determined by the last time an allocation change is made;
- Π: the number of communications, which depends on the passed messages summed over all UAVs during the consensus phases in Section 3.2 and the secondary conflict resolution in Section 3.3.2.;
- Ψ: the percentage of *feasible tasks* that are allocated to UAVs and the given time window constraints are satisfied.

The above parameter G reflects the objective of MATP, Λ determines the computation time of the whole algorithm, Π implies the communication cost of performing DATW, and Ψ illustrates the effectiveness of DATW in handling the time window constraints.

In this section, all experiments are conducted based on the scenarios summarized in Table 4. For each combination in Table 4, 30 instances are created under randomly generated initialization settings. A mesh communication topology is used such that each UAV can communicate with every other UAV. The simulation results are presented in Table 5.

Table 5. Result of the DATW Method in Search and Rescue Scenario.

UAV Num	Task Num	TUR	Ave. Completion Time (G)	Ave. Iteration Times (Λ)	Ave. Communication Num (Π)	Percentage of Feasible Tasks (Ψ)
3	3	1	103.88	1.27	5.83	100.0%
	6	2	149.05	2.47	9.07	93.8%
	9	3	149.02	3.13	11.03	74.1%
4	4	1	128.01	2.2	13.5	96.7%
	8	2	143.66	2.87	16.53	92.5%
	12	3	146.02	3.67	16.43	75.6%
9	9	1	110.37	3.03	42.4	99.3%
	18	2	132.89	5.1	67.73	96.7%
	27	3	139.16	6.4	65.47	84.7%
10	10	1	104.42	3.67	58.83	99.3%
	20	2	129.91	5.8	86.9	96.0%
	30	3	139.31	6.3	72.33	85.8%
15	15	1	100.87	4.8	114.23	98.4%
	30	2	123.82	6.83	147.37	97.0%
	45	3	135.28	8.07	142.27	87.7%
16	16	1	100.64	4.87	130.07	99.8%
	32	2	124.27	6.8	156.5	96.4%
	48	3	135.42	7.93	160.3	86.6%

According to Table 5, the average rate of feasible tasks exceeds 70% for each combination, which means that DATW is capable of finding a solution that allocates most tasks to UAVs properly. Moreover, we observe that the parameter TUR directly affects the average completion time and the number of iterations. In particular, the average completion time G logically rises with TUR as there are more tasks required to be performed. Additionally, as TUR becomes larger, each agent has increasingly available task options, such that the algorithm must iterate more times to select the valuable allocation. Hence, parameter Λ is enlarged. Moreover, the parameter Π increases with an increasing UAV number since more communications are required to reach consensus when the number of UAVs is increasing.

In addition, Π appears to take its minimum value at $\delta = 1$ when the number of UAVs is not changed. The reasons seem simple: the number of tasks is relatively small with $\delta = 1$, and hence, the allocated tasks are much easier to satisfy with the time window constraints. In this case, DATW may achieve its optimal allocation before the task reallocation phase.

4.3.2. Comparisons on the Task Reallocation Phase

Task reallocation, the third phase of DATW, aims to allocate the tasks to UAVs that are unassigned after the first two phases. To validate the effectiveness of this reallocation phase, we carry out our experiments to compare and analyze the allocations before and after this phase. Herein, each scenario combination in Table 4 is simulated 50 times under random initialization settings.

From Figure 3, when the TUR is set as $\delta \in \{1, 2, 3\}$, the line graphs depict the average number of allocated tasks before and after the task reallocation phase. The bar graphs depict the average increase in the number of allocated tasks during that phase. Notably, the task reallocation phase can significantly improve the total number of allocated tasks from the solution achieved by the first two phases with various UAV numbers and TURs. Comparing (a), (b), and (c) in Figure 3, for combinations with the same UAV numbers, more tasks are allocated during the third phase as the value of TUR increases. This is because when the value of TUR grows, there are more tasks involved, such that the time window constraints between tasks are more likely to overlap. As this situation leads to greater difficulties in assigning tasks, the task reallocation phase is necessary for obtaining a well-performed allocation. Moreover, we can observe that the number of tasks allocated through the third phase logically increases with the number of UAVs for each value of TUR. In the best case, the average increase in the number of allocated tasks is up to 8.04 when allocating 45 tasks to 15 UAVs.

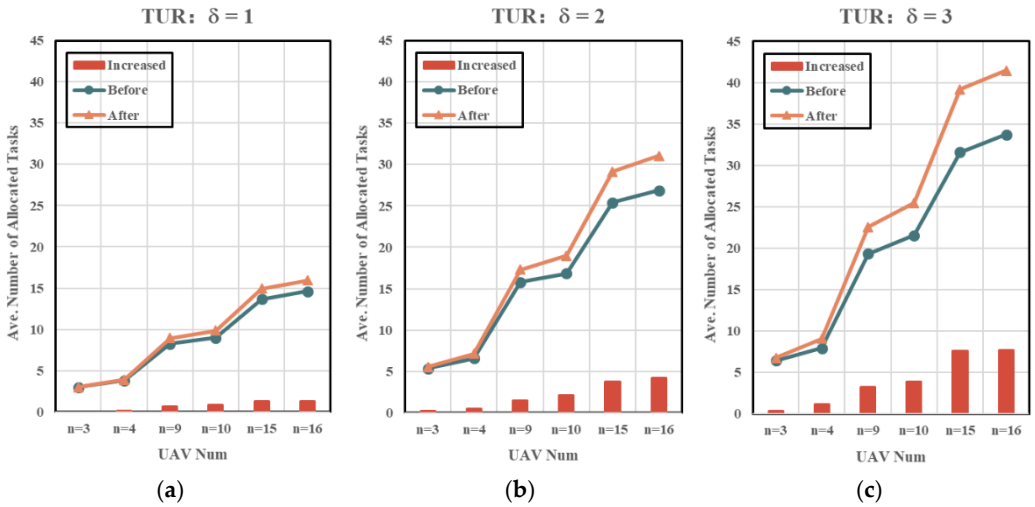

Figure 3. Average number of the allocated tasks before and after the task reallocation phase with regards to various TURs. (**a**) TUR: $\delta = 1$. (**b**) TUR: $\delta = 2$. (**c**) TUR: $\delta = 3$.

4.3.3. E-CBBA Versus DATW

Since the proposed DATW and the E-CBBA algorithm in [26] both aim to allocate tasks within validity time windows, we conduct a range of comparisons to determine the merit of the assignments achieved. Herein, we assume that the obtained allocation deems to be a success if each task is assigned to a proper UAV and complies with time window constraints. Then, based on this assumption, we introduce a new parameter, namely, the success rate (SR), to assess the performance of these two algorithms. All comparisons are conducted based on the *search and rescue* scenarios, where the UAV number is varied as a parameter from two to thirty and TUR is set as $\delta = 2$. Each combination of these scenarios

is simulated 50 times under random initialization settings, such that the success rate of a combination is calculated as follows:

$$SR = N_{sa}/50 \tag{17}$$

where N_{sa} denotes the number of successful allocations over all simulations and 50 is the total number of simulations.

Figure 4 compares the success rate of the DATW and the E-CBBA with different numbers of UAVs. The line graphs show the success rate of the two algorithms, while the bar graphs show the increase in success rate achieved by the DATW. Obviously, it demonstrates that the DATW consistently outperforms the E-CBBA algorithm in terms of success rate. Specifically, the average success rate for DATW is 52.7%, while the average success rate for E-CBBA is 34.7%, which means that the proposed DATW achieves an 18% (=52.7% − 34.7%) increase in the average success rate. Notably, in the combination of 22 UAVs and 44 tasks, all tasks are successfully allocated within validity time windows 32 times by DATW, but only 10 times by E-CBBA.

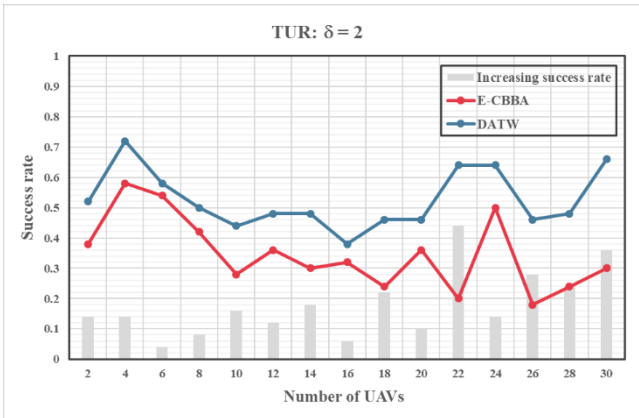

Figure 4. Success rate of allocations comparing E-CBBA and DATW algorithms.

4.3.4. Topology Comparison

The dynamic environment with moving UAVs incurs various changes in communication typologies between UAVs, and thus, the proposed algorithm is required to perform effectively in different communication topologies. Figure 5 depicts four basic communication topologies with nondirected graphs: a fully connected mesh topology, a row topology, a circular topology, and a star topology. Specifically, we aim to analyze how these different typologies affect the performance of DATW based on a *search and rescue* scenario and assess the DATW with respect to the number of communications (Π) and the percentage of *feasible tasks* (Ψ). All experiments are conducted on a scenario combination with 16 UAVs and 32 tasks, where the combination is simulated 50 times under random initialization settings.

The scatterplot in Figure 6 shows the result of over 50 simulations, which compares the PI, E-CBBA, and DATW in terms of parameters Π and Ψ. Obviously, the parameter Ψ is independent of the specific topology, while the parameter Π appears to vary according to the type of topology. In particular, the circular topology and the star topology have a similar degree of communication, such that Π and Ψ are similar across topologies for PI, E-CBBA, and DATW. Moreover, for each algorithm, the parameter Π decreases notably with fully connected mesh topology and increases with the row topology. Such an increase in Π is due to information requiring multiple communications among UAVs to reach a consensus when the network is not fully connected. Moreover, the PI algorithm cannot effectively handle the task allocation problem under various time window constraints, while the proposed DATW based on PI always achieves well-performing allocations. Additionally, we observe that under each communication topology, the distributions of the parameter Π are always similar, in terms of the DATW and PI algorithms. That is, the DATW achieves superior

allocations without imposing any extra communication burdens and allocates the most tasks to UAVs within time window constraints.

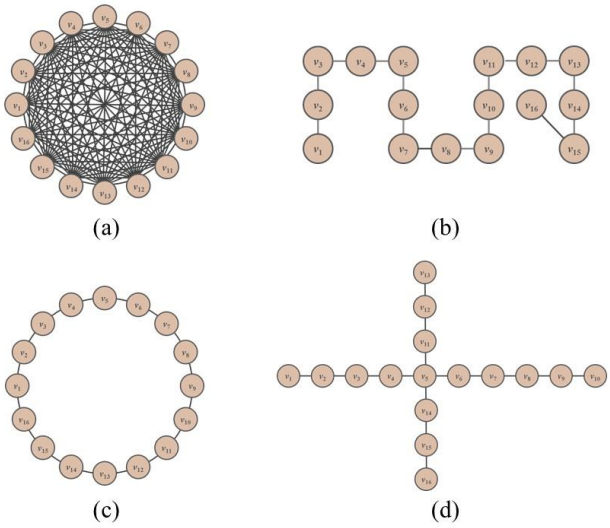

Figure 5. Four basic communication topologies. (**a**) Mesh communication. (**b**) Row communication. (**c**) Circular communication. (**d**) Star row communication.

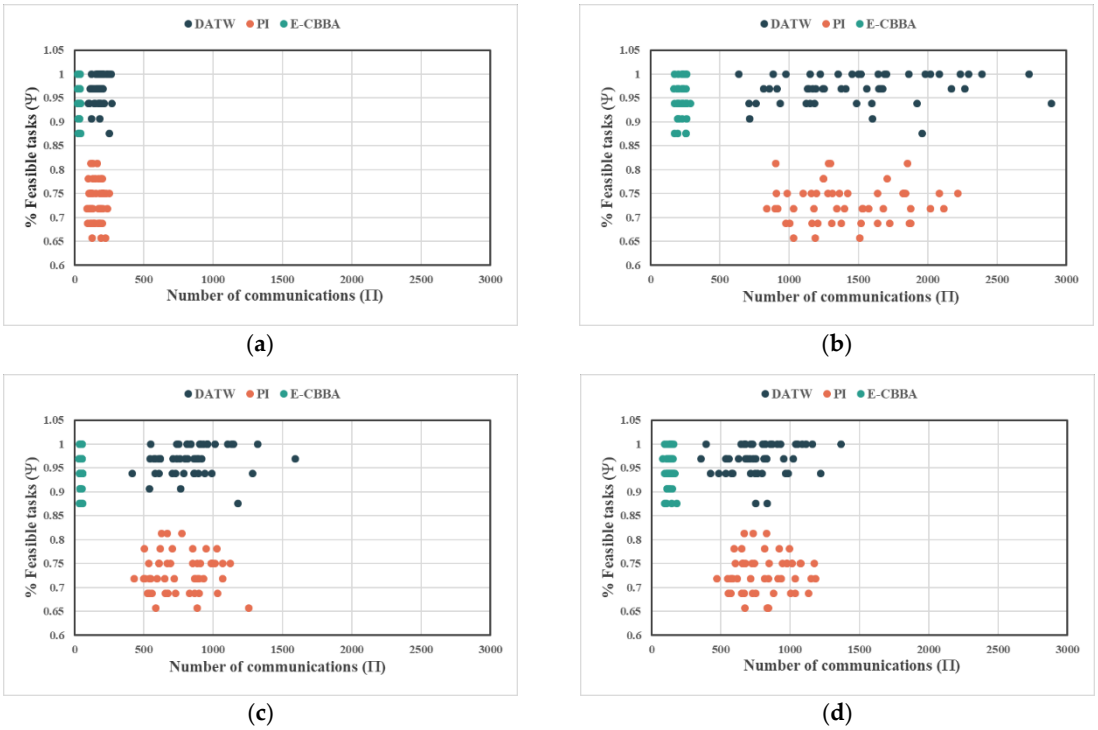

Figure 6. Scatterplot with respect to number of feasible tasks and number of communications over different network topologies. (**a**) Mesh communication. (**b**) Row communication. (**c**) Circular communication. (**d**) Star row communication.

5. Conclusions

In this paper, a distributed task allocation algorithm (DATW) is proposed for the MTAP problem with time window constraints. This algorithm embeds the time window constraints into the PI-based decentralized framework for the first time and aims to minimize the average completion time of all tasks. As an extension of the PI algorithm, the DATW method uses a three-phase architecture, including the task inclusion phase, conflict

resolution phase, and task reallocation phase. Herein, the average number of allocated tasks exceeds 70% by the aid of the newly introduced task reallocation phase. Another novel idea is to correlate the significance value of tasks with the time window constraints, such that each task can be completed as early as possible during its validity interval. Moreover, the start time of each task is broadcasted among agents via communication typology to allocate tasks in a validity time interval without imposing any extra communication burdens. To this end, a conflict-free allocation with a minimum average task completion time can be obtained by the proposed DATW algorithm, where the vast majority of tasks are assigned within the validity time window. A series of simulations are conducted under a search and rescue scenario. The experimental results confirm that the DATW is effective and superior in increasing the total number of allocated tasks as well as minimizing the average completion time of the tasks. Moreover, compared with existing (CBBA-based) solutions, results show up to an 18% increase in success rate (SR) using the DATW. Thus, the DATW can successfully enable a team of UAVs to perform complex missions under various time window constraints.

Future work will focus on facilitating real-time task allocation in dynamic environments, which indicates integrating the presented DATW into the asynchronous framework and addressing the MTAP with latency in task computation and communication.

Author Contributions: Conceptualization, W.C., R.L., Y.F. and Y.Y.; methodology, W.C. and R.L.; validation, W.C.; resources, Y.F. and Y.Y.; data curation, W.C.; writing—original draft preparation, W.C.; writing—review and editing, Y.F. and Y.Y.; supervision, Y.F. and Y.Y.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence", grant number 2020AAA0108200.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Shi, M.; Qin, K.; Liang, J.; Liu, J. Distributed Control of Uncertain Multiagent Systems for Tracking a Leader with Unknown Fractional-Order Dynamics. *Int. J. Robust Nonlinear Control* **2019**, *29*, 2254–2271. [CrossRef]
2. Shi, M.; Qin, K.; Liu, J. Cooperative Multi-Agent Sweep Coverage Control for Unknown Areas of Irregular Shape. *IET Control Theory Appl.* **2018**, *12*, 1983–1994. [CrossRef]
3. Shi, M.; Qin, K.; Li, P.; Liu, J. Consensus Conditions for High-Order Multiagent Systems with Nonuniform Delays. *Math. Probl. Eng.* **2017**, *2017*, 7307834. [CrossRef]
4. Li, X.; Li, C.; Yang, Y.; Mo, L. Consensus for Heterogeneous Multi-Agent Systems with Nonconvex Input Constraints and Nonuniform Time Delays. *J. Frankl. Inst.-Eng. Appl. Math.* **2020**, *357*, 3622–3635. [CrossRef]
5. Li, X.; Li, C.; Yang, Y. Heterogeneous Linear Multi-Agent Consensus with Nonconvex Input Constraints and Switching Graphs. *Inf. Sci.* **2019**, *501*, 397–405. [CrossRef]
6. Li, X.; Gao, K.; Lin, P.; Mo, L. A Further Result on Consensus Problems of Second-Order Multi-Agent Systems with Directed Graphs, a Moving Mode and Multiple Delays. *ISA Trans.* **2017**, *71*, 21–24. [CrossRef]
7. Yu, X.; Gao, X.; Wang, L.; Wang, X.; Ding, Y.; Lu, C.; Zhang, S. Cooperative Multi-UAV Task Assignment in Cross-Regional Joint Operations Considering Ammunition Inventory. *Drones* **2022**, *6*, 77. [CrossRef]
8. Bethke, B.; Valenti, M.; How, J.P. UAV task assignment. *IEEE Robot. Autom. Mag. March.* **2008**, *15*, 39–44. [CrossRef]
9. Chen, Y.; Yang, D.; Yu, J. Multi-UAV Task Assignment With Parameter and Time-Sensitive Uncertainties Using Modified Two-Part Wolf Pack Search Algorithm. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 2853–2872. [CrossRef]
10. Shima, T.; Rasmussen, S.J.; Sparks, A.G.; Passino, K.M. Multiple Task Assignments for Cooperating Uninhabited Aerial Vehicles Using Genetic Algorithms. *Comput. Oper. Res.* **2006**, *33*, 3252–3269. [CrossRef]
11. Darrah, M.; Niland, W.; Stolarik, B. Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission. In Proceedings of the Infotech@Aerospace, Arlington, VA, USA, 26 September 2005; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2005.
12. Stuijk, S.; Basten, T.; Geilen, M.C.W.; Corporaal, H. Multiprocessor Resource Allocation for Throughput-Constrained Synchronous Dataflow Graphs. In Proceedings of the 44th annual Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 777–782.

13. Secrest, B.R. *Traveling Salesman Problem for Surveillance Mission Using Particle Swarm Optimization*; School of Engineering and Management, Air Force Institute of Technology Wright-Patterson AFB: Dayton, OH, USA, 2001.
14. O'Rourke, K.P.; Carlton, W.B.; Bailey, T.G.; Hill, R.R. Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search. *Mil. Oper. Res.* **2001**, *6*, 5–30. [CrossRef]
15. Deng, W.; Zhang, X.; Zhou, Y.; Liu, Y.; Zhou, X.; Chen, H.; Zhao, H. An Enhanced Fast Non-Dominated Solution Sorting Genetic Algorithm for Multi-Objective Problems. *Inf. Sci.* **2022**, *585*, 441–453. [CrossRef]
16. Blum, C. Ant Colony Optimization: Introduction and Recent Trends. *Phys. Life Rev.* **2005**, *2*, 353–373. [CrossRef]
17. Wang, D.; Tan, D.; Liu, L. Particle Swarm Optimization Algorithm: An Overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
18. Sun, Y.; Wu, H.; Zhan, R. Optimal Scheduling Method of Marine Transportation Resources Based on Wolf Swarm Algorithm. *J. Coast. Res.* **2019**, *93*, 646–651. [CrossRef]
19. Gerkey, B.P.; Mataric, M.J. Sold!: Auction Methods for Multirobot Coordination. *IEEE Trans. Robot. Autom.* **2002**, *18*, 758–768. [CrossRef]
20. Qin, B.; Zhang, D.; Tang, S.; Wang, M. Distributed Grouping Cooperative Dynamic Task Assignment Method of UAV Swarm. *Appl. Sci.* **2022**, *12*, 2865. [CrossRef]
21. Lemaire, T.; Alami, R.; Lacroix, S. A Distributed Tasks Allocation Scheme in Multi-UAV Context. In Proceedings of the IEEE International Conference on Robotics and Automation, Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 26 April–1 May 2004; Volume 4, pp. 3622–3627.
22. Oh, G.; Kim, Y.; Ahn, J.; Choi, H.-L. Market-Based Distributed Task Assignment of Multiple Unmanned Aerial Vehicles for Cooperative Timing Mission. *J. Aircr.* **2017**, *54*, 2298–2310. [CrossRef]
23. Zhen, Z.; Wen, L.; Wang, B.; Hu, Z.; Zhang, D. Improved Contract Network Protocol Algorithm Based Cooperative Target Allocation of Heterogeneous UAV Swarm. *Aerosp. Sci. Technol.* **2021**, *119*, 107054. [CrossRef]
24. Choi, H.-L.; Brunet, L.; How, J.P. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Trans. Robot.* **2009**, *25*, 912–926. [CrossRef]
25. Whitten, A.K.; Choi, H.-L.; Johnson, L.B.; How, J.P. Decentralized Task Allocation with Coupled Constraints in Complex Missions. In Proceedings of the 2011 American Control Conference, San Francisco, CA, USA, 29 June–1 July 2011; pp. 1642–1649.
26. Ponda, S.; Redding, J.; Choi, H.-L.; How, J.P.; Vavrina, M.; Vian, J. Decentralized Planning for Complex Missions with Dynamic Communication Constraints. In Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 3998–4003.
27. Whitbrook, A.; Meng, Q.; Chung, P.W.H. A Novel Distributed Scheduling Algorithm for Time-Critical Multi-Agent Systems. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 6451–6458.
28. Zhao, W.; Meng, Q.; Chung, P.W.H. A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario. *IEEE Trans. Cybern.* **2016**, *46*, 902–915. [CrossRef] [PubMed]
29. Geng, L.; Zhang, Y.F.; Wang, J.; Fuh, J.Y.; Teo, S.H. Cooperative Mission Planning with Multiple UAVs in Realistic Environments. *Unmanned Syst.* **2014**, *2*, 73–86. [CrossRef]
30. Zheng, X.; Zhang, F.; Song, T.; Lin, D. Heterogeneous Multi-UAV Distributed Task Allocation Based on CBBA. In Proceedings of the 2019 IEEE International Conference on Unmanned Systems (ICUS), Beijing, China, 17–19 October 2019; pp. 704–709.
31. Turner, J.; Meng, Q.; Schaefer, G.; Whitbrook, A.; Soltoggio, A. Distributed Task Rescheduling With Time Constraints for the Optimization of Total Task Allocations in a Multirobot System. *IEEE Trans. Cybern.* **2018**, *48*, 2583–2597. [CrossRef] [PubMed]