Hindawi
International Journal of Aerospace Engineering
Volume 2020, Article ID 8878136, 15 pages
https://doi.org/10.1155/2020/8878136

*Research Article*

# A Damage-Tolerant Task Assignment Algorithm for UAV Swarm in Confrontational Environments

**Chao Chen, Weidong Bao, Tong Men, Wen Zhou, Daqian Liu, and Li Ma**

*College of Systems Engineering, National University of Defense Technology, Changsha Hunan 410073, China*

Correspondence should be addressed to Weidong Bao; wdbao@nudt.edu.cn

Received 2 June 2020; Revised 27 July 2020; Accepted 11 August 2020; Published 28 August 2020

Academic Editor: Yancheng You



As Unmanned Aerial Vehicles (UAVs) are widely used in many applications, a lot of military missions in confrontational environments are being undertaken by UAV swarm rather than human beings due to its advantages. In confrontational environments, the reliability and availability of UAV swarm would be the major concern because of UAVs' vulnerability, so damage-tolerant task assigning algorithms are of great importance. In this paper, we come up with a novel damage-tolerant framework for assigning real-time tasks to UAVs with dynamical states in confrontational environments. Different from existing scheduling methods, we not only assign tasks but also back up copies of tasks to UAVs when needed, to promote reliability. Meanwhile, we adopt an overlapping mechanism, including Backup-Primary overlapping and Backup-Backup overlapping, in assignment to save the limited swarm resources. On the basis of the damage-tolerant and overlapping mechanism, for the first time, we propose a new damage-tolerant task assignment algorithm named DTTA, aiming at promoting the task success probability. Extensive experiments are conducted based on random synthetic workloads to compare DTTA with three baseline algorithms. The experimental results indicate that DTTA can efficiently promote the probability of tasks' success without affecting the effectiveness of swarms in confrontational environments.

## 1. Introduction

With the development of mechanical technologies, automation technology, and artificial intelligence, Unmanned Aerial Vehicles (UAVs) are becoming more and more capable in various industries. Owing to its low cost, no risk of casualties, and good maneuverability, it has been widely applied in many scenarios to replace manual tasks in recent years, e.g., remote sensing [1], precision agriculture [2], disaster relief [3], and logistics service [4]. Especially in military operations, UAVs are replacing warriors to undertake more and more work [5], showing its great advantages in concealment, survivability, and diversified capabilities. Moreover, the large-scale use of UAVs, named swarm, has brought new ways of generating fighting power [6].

In the applications of UAV swarms, how to assign tasks is a key factor that greatly influences the performance. And the main problem of the assignment process is how to effectively allocate arrived tasks to proper UAVs, with the consideration of task demands, time constraints, capability requirements, and UAV real-time states [7]. Besides, the complexity of UAV swarm management would grow dramatically with the increase of scale. That is because the amount of computation for task assignment optimization increases exponentially with the number of UAVs. Especially in the battlefield [8, 9], the damaging of UAVs, the weak communication condition, and uncertainty environment [10] should not be ignored. It should be noted that the damages considered in this paper contain all kinds of abnormal conditions which would lead to the failure of task execution. Thus, the task assignment process in the confrontational environment would face two main challenges apart from regular constraints: one is that UAVs may be damaged, and the tasks on it should be reallocated; another is that the real-time requirements and weak communication conditions would make rescheduling after the occurrence of damaging infeasible.

As the real-time reallocation method is not feasible, another solution is the task assigning plan which can tolerate UAV damaging problem; that is to say, the scheduling plan should ensure the tasks on a damaged UAV to be finished

successfully as many as possible, which can be achieved by using multiple copies of tasks on different UAVs. The damage-tolerant mechanism is derived from the studies on fault-tolerant scheduling, among which the primary and backup model is the most used scheme [11].

However, the problem of damage-tolerant for UAV swarm has distinct differences: the communication channel is usually unstable, the tasks on UAVs usually have different locations, and the heterogeneous UAVs usually have diverse capabilities. Such features make it hard to study the fault-tolerant mechanism in UAV swarms.

To the best of our knowledge, few studies have a focus on the damage-tolerant problem of UAV swarm applications in confrontational environments. And the traditional fault-tolerant assignment algorithms cannot well fit the confrontational UAV swarm scenario. Consequently, we attempt to propose a damage-tolerant mechanism on the base of traditional fault-tolerant methods and give damage-tolerant assigning algorithms for UAV swarms in confrontational environments. The main contributions of this study are concluded below:

(i) We designed a new damage-tolerant assignment framework, including damage-tolerant model and mechanisms

(ii) We studied the overlapping model for resource saving, and the properties of the model are analyzed in detail

(iii) We proposed a damage-tolerant algorithm (DTTA) for task assignment on UAV swarms in confrontational environments, which includes Backup-Primary (BP) overlapping and Backup-Backup (BB) overlapping mechanism

(iv) We conducted extensive simulation experiments to verify the effectiveness of the proposed damage-tolerant mechanism and algorithms

The rest of the paper is organized as follows. In Section 2, we summarize the related work in the literature. The definitions and damage-tolerant models are given in Section 3. And Section 4 illustrates the damage-tolerant mechanism. In Section 5, a damage-tolerant task assigning algorithm—DTTA—is given in detail. This is followed by the experiments and performance analysis in Section 6. Finally, we summarize the research and future directions in Section 7.

## 2. Related Works

In this section, we will summarize the related literature from three corresponding aspects: assigning algorithm, damage-tolerant mechanism, and overlapping mechanism.

For assigning algorithms, many researchers have developed effective strategies for cooperating UAVs. Generally, scheduling algorithms can be divided into two categories: centralized methods and distributed ones. As discussed by Shumacher [12], a centralized UAV task scheduling method was studied, and the authors solved the optimization problem by using a mixed-integer linear programming method. Liu and Kroll [13] aimed to solve the multirobot task allocation problems, and they developed a subpopulation-based genetic algorithm by using inversion mutation and selection. Other centralized algorithms like ant colony optimization [14], genetic algorithm [15], and wolf pack algorithm [16] also show effectiveness in solving the task allocation problem for multirobot systems. Though the centralized algorithms show advantages in optimization, the huge real-time communication pressure of the center and complexity of computation limit its performance. And the decentralized methods are becoming increasingly popular when facing dynamic tasks and uncertain environment. Kim et al. [17] developed a distributed task allocation method based on their proposed measure of resource welfare. The method can acquire a globally high resource welfare and quick attack response for the UAV team by utilizing limited resources in a balanced way. As discussed in [7], the authors aimed to produce a satisfactory solution in a reasonable amount of time rather than finding the global optimal solution in an unacceptable amount of time presented, and they proposed a new hierarchical method to solve the task assignment problem for multiple UAV teams, which divided the assignment process into two phases: clustering and assigning clusters. The experiment results identified the effectiveness of its application to intelligent robots. Besides, there are many other models such as contract net (CN) [18], decentralized Markov decision process (Dec-MDP) [19], market auction mechanism [20], and dynamic ant colony's labor division (DACLD) [21]. However, for the battlefield dynamic tasks in confrontational environments, these models and algorithms are weak in some points; they did not take the damaging problem of UAVs and heterogeneous UAVs into account when doing task allocation.

For the damage-tolerant mechanism, there is very little research about it. However some literature related to fault-tolerant mechanisms of UAVs can be found, which is similar to damage-tolerant methods. And the researchers mainly focus on fault-tolerant control mechanisms in the applications of UAVs [22, 23]. In [24], Yu et al. proposed a fault-tolerant control solution against the actuator faults for UAVs. And the fault scenarios happen on the actuator, so the fault-tolerant method only could be used to deal with a fault in behaviour level. In order to solve more complicated fault-tolerant problems, a multilevel reconfiguration framework was proposed for UAVs in [25], and the authors tried to accommodate fault problems of different levels. By the demonstration on UAV based on a full six-degrees-of-freedom nonlinear simulation model, the effectiveness was identified. It is worth noting that the fault control mechanisms only focus on the problems of single UAV. As for UAV swarms, the fault tolerance mechanism was not studied yet.

In this paper, we would adopt the replication technology to achieve damage tolerance for UAV swarms. Based on the replications, the assignment algorithms try to assign the copies of tasks to enhance reliability. A mainly used fault-tolerant model is the primary-backup model [11, 26], where the tasks would have primary copies and backup copies in case of encountering failure. By using replications, the backup copies would consume much more resources, leading

to the poor efficiency of UAVs. In order to reduce the extra resource expense of backups, the overlapping mechanism was studied by many researchers. By using the overlapping model, the backups can overlap other tasks for resource saving. The backup overlapping for dependent tasks was investigated in [27], and the authors designed a fault-tolerant scheduling algorithm in heterogeneous systems. And as discussed elsewhere [28, 29], an overlapping model is also identified to be an effective mechanism to promote the utilization of resources.

To the best of our knowledge, few previous works have researched the assigning algorithm, damage-tolerant mechanism, and overlapping model collaboratively. Consequently, we propose DTTA to promote the successful probability of tasks in confrontational environments with no damage to UAVs' efficiency.

## 3. Definitions and System Model

In this section, we firstly introduce the notations used in this article. Then, we present task models and UAV models. Furthermore, the task assignment framework for smart swarm consisting of UAVs is to be illustrated. Finally, we would introduce assigning objectives.

*3.1. Main Notations.* In Table 1, we summarize the main notations adopted in this paper. It is worth noting that $\mathrm{et}_i$ is a decimal fraction used to indicate the risk level faced by the executing UAV, and the value of $\mathrm{et}_i$ is usually between 0 and 1. The value of time is simulated by a positive real number, we set the initial moment to be 0, and the unit of the time is seconds. For example, if the value of $\mathrm{ta}_i$ is 5, that means the task will arrive in the moment of 5 s; if the value of $\mathrm{td}_i$ is 10.2, that means the task should be finished by 10.2 s at the latest; if the value of $\mathrm{en}_j$ is 1500, that means the UAV's flying can last for 1500 seconds; if the value of $\mathrm{lft}_j$ is 36, that means UAV $u_j$ would be busy until 36 s to finish its task list; if the value of $e_i$ is 3, that means the UAV $u_j$ should spend 3 seconds on the execution of task $t_i$. If the value of $\mathrm{pt}_{ij}$ is 10, that means the UAV needs to spend 10 seconds to be prepared for the execution of task $t_i$ after finishing $u_j$'s last task in its task list.

*3.2. Task Model and UAV Model.* In this paper, we choose three kinds of representative tasks, which are the reconnaissance task [30], striking task [31], and damage assessment task [32], respectively, to study the damage-tolerant task assignment algorithm. And we assume that all UAVs would take off at the base, and each UAV's departure time is determined by the moment the first task of it is assigned.

We use a set $T = \{t_1, t_2, t_3, \cdots, t_n\}$ to denote the real-time tasks which arrive aperiodically. As the tasks to be assigned are imported by the commander, the coupling tasks' decomposition process can be handled; thus, we assume the tasks to be scheduled are independent. In addition, the UAV resources on the battlefield would face complex confrontational environments and weak communication environment. It would be harder to schedule tasks in a preemptive mode, which needs reliable real-time state parameters. Therefore, the tasks in set $T$ are non-preemptive and independent to simulate the UAV tasks in confrontational environments.

We model task $t_i \in T$ as a tuple $t_i = t_i = (\mathrm{ty}_i, \mathrm{ta}_i, \mathrm{td}_i, p_i, \mathrm{e}\,\mathrm{t}_i, \mathrm{si}_i)$, where $\mathrm{ty}_i$, $\mathrm{ta}_i$, $\mathrm{td}_i$, $p_i$, $\mathrm{et}_i$, and $\mathrm{si}_i$ represent task $t_i$'s task type, arrival time, deadline, task execution position, task execution environment-threatened value, and supplementary instruction, respectively. It is worth noting that the task type can be reconnaissance, striking, or damage assessment task.

Let $U = \{u_1, u_2, u_3, \cdots, u_m\}$ be a UAV set. UAV $u_j$ is modeled as a tuple $u_i = (\mathrm{en}_j, v_j, \mathrm{at}_j, \mathrm{lft}_j, \mathrm{ltp}_j, \mathrm{rsr}_j, \mathrm{tft}_j)$, where $\mathrm{en}_j$, $v_j$, $\mathrm{at}_j$, $\mathrm{lft}_j$, $\mathrm{ltp}_j$, $\mathrm{rsr}_j$, and $\mathrm{tft}_j$ denote UAV $u_j$'s endurance time, flying velocity, ability type set, finish time of the last assigned task, position of the last assigned task, remained striking resources, and total flying time, respectively. It should be noted that the ability type can be reconnaissance ability or striking ability and damage assessment ability. The element $\mathrm{rsr}_j$ represents the striking resources remained in $u_j$; when the value of $\mathrm{rsr}_j$ is equal to 0, that means UAV $u_j$ can no longer undertake striking tasks. In addition, $\mathrm{tft}_j$ is used to indicate the total flying time of $u_j$, when UAV $u_j$ is damaged in the confrontational environments, $\mathrm{tft}_j$ represents the time from its takeoff to the finish moment of $u_j$'s last successfully executed task; otherwise, $\mathrm{tft}_j$ is the time from its takeoff to landing.

There are many other parameters related to both UAVs, and tasks should be illustrated in detail. Take Figure 1 as an example: $t_k$ is the last task to be processed in UAV $u_j$'s task list, and $t_i$ is a task to be assigned. We use $\mathrm{pt}_{ij}$ to denote the preparing time of UAV $u_j$ to execute task $t_i$, which mainly refers to the flying time from position $p_k$ to position $p_i$. It is worth noting that the $\mathrm{pt}_{ij}$ would refer to flying time from UAV base position to task $t_i$ execution position when there is no task that has been assigned to UAV $u_j$. We use $\mathrm{sp}_{ij}$ and $\mathrm{se}_{ij}$ to represent the preparation start time and execution start time of task $t_i$ on UAV $u_j$, of which $\mathrm{sp}_{ij}$ means the start time of the preparing process. The parameter $\mathrm{st}(t_i^B)$ is used to denote the status of the backup task $t_i^B$, and the value of it will be illustrated in part 4. The $\mathrm{lft}_j$ is the finish time of the last task that has been allocated to UAV $u_j$; take Figure 1 as an example: the value of $\mathrm{lft}_j$ equals to $f_{kj}$ before task $t_i$ is assigned while the value is equal to $f_{ij}$ after the assignment. In addition, the relationship between these parameters in the assigning process can be described as equation (1), where the parameters can be found in Table 1.

$$\begin{cases} \mathrm{sp}_{ij} \geq \mathrm{ta}_i, \\ \mathrm{se}_{ij} = \mathrm{sp}_{ij} + \mathrm{pt}_{ij}, \\ f_{ij} = \mathrm{se}_{ij} + e_{ij} \leq \mathrm{td}_i. \end{cases} \tag{1}$$

*3.3. Task Assignment Framework.* In a confrontational combat environment, various threat factors (enemy firepower, high electromagnetic environment, harsh natural environment, etc.) are easy to make the missions of the UAVs fail.

Table 1: Definition of main notations.

| Notation | Definition |
|---|---|
| $t_i$ | The $i^{\text{th}}$ task in task set |
| $\text{ty}_i$, $\text{ta}_i$, $\text{td}_i$, $p_i$, $\text{et}_i$, $\text{si}_i$ | Task $t_i$'s type, target appearance time, deadline, execution position, environment-threatened value, supplementary instruction |
| $u_j$ | The $j^{\text{th}}$ UAV in UAV set |
| $\text{en}_j$, $v_j$, $\text{at}_j$, $\text{lft}_j$, $\text{ltp}_j$, $\text{rsr}_j$, $\text{tft}_j$ | UAV $u_j$'s endurance time, maximum velocity, ability type, finish time of the last execution task, position of the last execution task, remained striking resource, total flying time |
| $\text{pt}_{ij}$ | The preparation time of UAV $u_j$ to execute task $t_i$ |
| $\text{EFT}_j(t_i)$ | The earliest finish time of task $t_i$ on UAV $u_j$ |
| $e_{ij}$, $f_{ij}$ | The execution time and finish time of $t_i$ on $u_j$ |
| $\text{sp}_{ij}$, $\text{se}_{ij}$ | The preparation start time and execution start time of task $t_i$ on UAV $u_j$ |
| $\text{st}(t_i)$ | $\text{st}(t_i)$ is "1" if task $t_i$ has backup $t_i^B$; otherwise, $\text{st}(t_i)$ is "0" |
| $U(t_i^P)$, $U(t_i^B)$ | The UAV to which $t_i^P$ and $t_i^B$ are assigned |
| $x_{ij}^P$, $x_{ij}^B$ | $x_{ij}^P(x_{ij}^B)$ is "1" if $t_i^P(t_i^B)$ is assigned to $u_j$; otherwise, $x_{ij}^P(x_{ij}^B)$ is "0" |

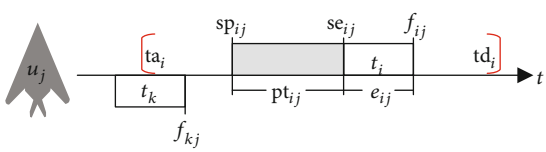


Figure 1: The task assigning parameters.

In order to improve the task success ratio, we designed a new task assignment framework, which is shown in Figure 2.

Different from other UAV task allocation frameworks [21], we use task backup to improve the success probability of tasks. As can be seen in Figure 2, the tasks are produced by the commanders, who will design the combat operations according to the combat purpose, satellite observation, and UAV reconnaissance. When a commander inputs the task into the assignment system, the task manager would analyze relative parameters and select available UAVs as candidates. Then, the task scheduler would assign the task's primary copy to a proper UAV, and the backup copy of the task, if needed, would be scheduled in the meantime. There are two main reasons we choose backup mechanism rather than real-time reallocation after damaging, one is that the available UAVs may be occupied by other tasks with loose time constraints before the time of damaging occurs, and another is that the weak communication condition cannot support the real-time reallocation's demands for real-time state parameters.

The backup controller needs to manage task backup copies. In the assignment process, whether a task's backup copy is needed or not is decided by the environment of the task, which is because the UAV in a more threatening environment has a higher probability to be damaged by the enemies; thus, a backup is needed. And in the managing process, the backup controller would cancel the backup $t_i^B$ when the primary copy $t_i^P$ has been executed successfully, and in other cases, the backup will not be canceled until its execution.

For the UAVs, they receive task information from the task assignment system and report their state information to the system. In addition, the UAVs will contribute the intelligence information to the commander for battlefield planning.

*3.4. Damage Model.* In our research, we focus on the battle damage of the UAVs in confrontational environments. If a UAV is damaged in the execution process, the tasks in its list would fail to finish. In our study, we apply a backup mechanism to battlefield tasks to promote the success probability of them.

In addition, the damaging detection mechanism like fail signal [28, 33] is existing in UAVs, reporting about UAV battle damages. Moreover, newly arrived tasks will not be assigned to a known damaged UAV.

As for the damaging probability in the case UAV $u_j$ executing task $t_i$, the value can be simulated by equation (2) according to the following analysis. The damaging probability value has a positive relationship with execution time while it has a negative relationship with velocity and environment-threatened value. And the parameter $\mu$ is used for linear adjustment.

$$p_{\text{damage}} = \frac{\mu e_{ij}}{v_j}\left(1 - \frac{1}{e^{\text{et}_i}}\right). \tag{2}$$

In this paper, we will first analyze the damaging of one UAV in the two related (primary and backup relation) UAVs. Then, we would extend the model to multi-UAV damages. Furtherly, the mechanism of a multidamage scenario will be discussed for damage tolerance.

The damaging model would ignore some nonsignificant factors about the actual scene. That is because our paper is an original innovation base research about the damage-tolerant problem for UAV in confrontational environments, and we focus on the mechanism and rules on the damage-

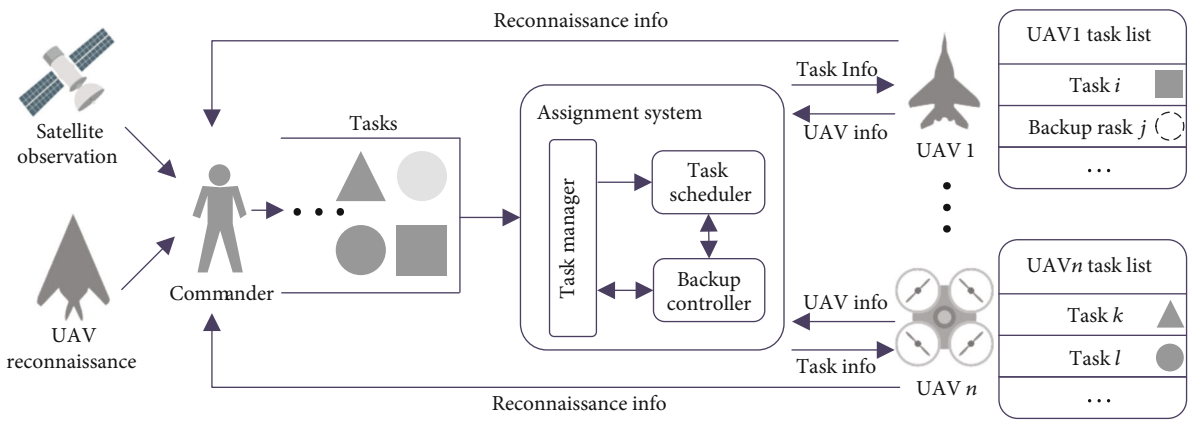


Figure 2: Task assignment framework.

tolerant problem. Further research will consider a more complex scenario and take more complicated factors into account.

*3.5. Assignment Objectives.* In our study, we pay attention to the task assignment process for UAVs in confrontational environments. The feature of such a scene is that the UAV may be damaged by enemies, making tasks easy to fail, and the UAV resources available are usually limited. Consequently, we take a backup algorithm into account to promote success probability. The main objectives are to accommodate as many battlefield tasks as possible and achieve higher utilization of heterogeneous UAVs. In addition, we study the backup overlap mechanism to save resources to make the utilization for UAVs as higher as possible in the case of ensuring task success.

The overarching assignment goal modeled in formula (3) is to maximize the number of successfully assigned tasks under timing constraints.

$$\max_{t_i \in T, u_j \in U} \left( \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}{}^{p} \right). \tag{3}$$

As for the utilization of UAVs, the higher the value is, the better. We delineate the goal in

$$\max_{t_i \in T, u_j \in U} \left\{ \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} e_{ij} * x_{ij}{}^{p}}{\sum_{j=1}^{m} \mathrm{tft}_j} \right\}. \tag{4}$$

The above equations indicate that our damage-tolerant assignment scheme has to aim at scheduling more tasks to UAVs for successful execution within complex constraints and realize higher resource utilization.

## 4. Damage-Tolerant Mechanism

To deal with the problem that UAVs may be damaged in a confrontational battlefield environment, the task backup mechanism is employed in our work for damage tolerance. The main idea of the backup mechanism is that an arrived task may have a backup copy, which is decided by the environment-threatened condition, in another UAV in case of damaging problem. In addition, the backup overlapping methods, which include Backup-Primary (BP for short) overlapping and Backup-Backup (BB for short) overlapping, are used to save UAV resources. That is because a considerable part of task backup copies only occupies UAV time slot but do not process as UAV damaging probability is not very high.

*4.1. BP Overlapping Model.* The results and discussion may be presented separately, or in one combined section, and may optionally be divided into headed subsections.

Take Figure 3 as an example to illustrate the backup and BP overlapping model. The backslash rectangle represents the backup copy of the task $t_a$. It should be noted that the preparation time of the primary copy and backup copy would be different; this is because the locations of the candidate UAVs are different. As depicted in Figure 3, the part where $t_a{}^B$ and $t_b{}^P$ coincide on $u_k$'s time axis is overlapping time.

As the settings before, we firstly analyzed the situation that at most one damage problem occurs in a certain time slot, so there would be four cases for the UAVs in Figure 3:

(i) If UAV $u_j$, $u_k$, and $u_l$ encounter no damage, $t_a{}^P$ will be processed successfully, and the backup copy $t_a{}^B$ will be canceled. In addition, a time slot of $u_k$ occupied by $t_a{}^B$ will be released, and task $t_b{}^P$ will be executed. Finally, $t_a$ and $t_b$ will be executed successfully and backup copies of them will be canceled, with no task conflict

(ii) If only UAV $u_j$ encounters a damage problem during the time slot, task $t_a{}^P$ will fail, leading to the situation that backup copy $t_b{}^B$ should be processed for damage tolerance. As a result, task $t_b{}^P$ will be canceled, and backup copy task $t_b{}^B$ will be executed. Finally, tasks $t_a$ and $t_b$ will be executed successfully in the situation of $u_j$'s being damaged, with no task conflict

(iii) If UAV $u_k$ encounters a damage problem during the time slot, task $t_b{}^P$ will fail, and task $t_a{}^P$ and task $t_b{}^B$ will be processed successfully. Finally, tasks $t_a$ and

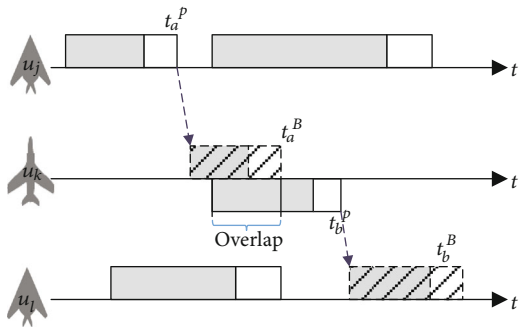


Figure 3: An example for BP overlapping.

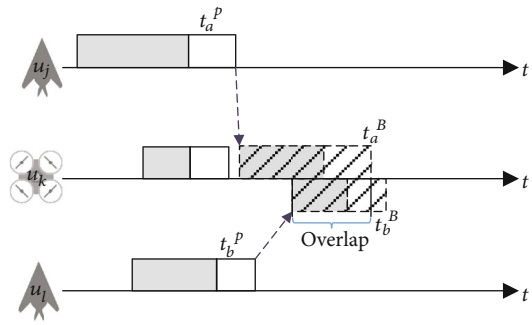


Figure 4: An example for BB overlapping.

$t_b$ will be finished in the situation of $u_k$'s being damaged, with no task conflict

(iv) If UAV $u_l$ encounters a damage problem, task $t_a{}^P$ and task $t_b{}^P$ will be finished successfully, and task $t_a{}^B$ and task $t_b{}^B$ will be canceled. In the situation of $u_k$'s being damaged, tasks $t_a$ and $t_b$ will be successful, with no task conflict

Furtherly, the more complex situation is that more than one damage problem occur. When $u_j$ and $u_k$ encounter damage problem in a certain time slot, tasks $t_a{}^P$, $t_b{}^P$, and $t_b{}^B$ would fail, but $t_a{}^B$ would succeed. Similarly, in other cases that only two of the three UAVs in Figure 3 encounter a damage problem, one of the two tasks would succeed. Only in the case that $u_j$, $u_k$, and $u_l$ encounter damaging at the same time in a certain time slot, the two tasks all fail. As is discussed above, the overlapping model can effectively promote task success probability with little extra UAV resource expense.

Consequently, the BP overlapping model is effective for damage tolerance.

*4.2. BB Overlapping Model.* Similar to the BP model, we give an example for the BB overlapping model in Figure 4. The backslash rectangle above and below the axis represents the backup copies of tasks $t_a$ and $t_b$, respectively. And the two backup tasks overlap in $u_k$'s time slot for damage-tolerant.

As for the case that at most one damage problem occurs, we consider the damaging problem of four conditions:

(i) If UAV $u_j$, $u_k$, and $u_l$ fly safely, tasks $t_a{}^P$ and task $t_b{}^P$ will be executed, and their backup copies will be canceled. In addition, no task conflict will occur

(ii) If UAV $u_j$ encounters a damage problem during the time slot, task $t_a{}^P$ will fail, and task $t_a{}^B$ and task $t_b{}^P$ will be executed. Finally, tasks $t_a$ and $t_b$ will succeed in the situation of $u_j$'s damaging, with no conflict

(iii) If UAV $u_k$ encounters a damage problem, task $t_a{}^P$ and task $t_b{}^P$ will be executed. Finally, tasks $t_a$ and $t_b$ will be finished in the situation of $u_k$'s being damaged, with no task conflict

(iv) If UAV $u_l$ encounters a damage problem, task $t_a{}^P$ and task $t_b{}^B$ will be finished successfully, and task $t_a{}^B$ and will be canceled. In the situation of $u_l$'s being damaged, tasks $t_a$ and $t_b$ will be successful, with no task conflict

And for the complex situation of the multidamage case, the analysis is similar to that in the part of BP overlapping.

Consequently, the backup copies can overlap for the consideration of UAVs' utilization, and the BB overlapping model is effective for damaging problems in confrontational environments.

*4.3. Basic Properties for Overlapping Model.* Based on the models, we firstly introduce some basic properties to indicate some constraints that the overlapping mechanism should satisfy.

*Property 1.* The UAV executing primary copy $t_a{}^P$ must be different from the UAV where the corresponding backup copy $t_a{}^B$ is allocated.

$$\forall t_a \in T, U\left(t_a^P\right) \neq U\left(t_a^B\right). \tag{5}$$

Property 1 suggests that the backup copy and primary copy should not be scheduled to the same UAV. That is because when a damage problem occurs on the UAV, both the backup copy and primary copy would be canceled, leading to the failure of the task. That is to say, the damage tolerance would not work under this condition.

*Property 2.* The preparation start time of backup copy $t_a{}^B$ should be later than the finish time of $t_a{}^P$ and the finish time of $U(t_a{}^B)$'s last execution task.

$$\forall t_a \in T, \mathrm{sp}_{aj}^B > f_{ak}^P, \mathrm{sp}_{aj}^B > \mathrm{lft}_j, \quad k \neq j. \tag{6}$$

Property 2 indicates that the backup copy and primary copy of the same task would have no overlapping on the timeline. The reason is that different tasks are usually in different locations; the start of a backup task would strongly affect the UAV's response time to the newly arrived tasks (it is scarcely possible to predict when the damaging would happen; thus, no UAV position can be used for assignment),

leading to the situation that a task occupies twice the UAV resources, which is not in line with our idea of achieving higher UAV efficiency. So we set the start time of backup copy to be later than the finish time of its primary copy. Another idea described in Property 2 is that the start time of backup copy should be later than the UAV's last execution task; it is because the resource cannot undertake two different tasks at the same time.

*Property 3.* Task $t_a$ can be assigned to UAV $u_j$ if and only if $u_j$'s ability types can satisfy $t_a$'s task-type requirement.

$$\begin{cases} \forall t_a \in T, x_{aj}^P = 1, & \mathrm{ty}_a \in \mathrm{at}_j, \\ \forall t_a \in T, x_{ak}^B = 1, & \mathrm{ty}_a \in \mathrm{at}_k. \end{cases} \tag{7}$$

Equation (7) suggests that the UAV's ability type must satisfy task-type demands of all tasks that have been assigned to it.

*Property 4.* Task $t_a$'s primary copy can overlap with other tasks if and only if task $t_a$ has backup copies.

In our work, not all tasks have backup copies in the consideration of resource efficiency. When $\mathrm{st}(t_a)$ is equal to 0, that means there is no backup copy for $t_a$, and the primary copy must be executed entirely on the undertaking UAV. Consequently, ${t_a}^P$ cannot overlap with other tasks.

*Property 5.* If a primary copy ${t_a}^P$ overlaps with a backup copy ${t_b}^B$ on UAV $u_j$, then the start preparation time $\mathrm{sp}_{bj}{}^B$ must be earlier than $\mathrm{sp}_{aj}{}^P$.

According to the overlapping mechanism, a backup copy task belongs to a kind of task that may be processed, and it must begin after the failure of its corresponding primary task. However, a primary copy task belongs to a kind of tasks that cannot be suspended unless damaging occurs. That is to say, once a primary copy task has started, it would occupy the UAV until its finish time. If $\mathrm{sp}_{bj}{}^B$ is later than $\mathrm{sp}_{aj}{}^P$, the backup copy is meaningless because task ${t_a}^P$ will occupy $u_j$ until its finishing no matter whether ${t_b}^P$ fails or not. Take Figure 5 as an example for illustration.

If $u_k$ encounters a damage problem, task ${t_b}^P$ would fail and backup copy task ${t_b}^B$ needs to be executed. However, task ${t_a}^P$ has occupied the UAV $u_j$, making task ${t_b}^B$ fails to start. As a result, the assignment plan is not able to tolerate the damaging problem. Therefore, the start preparation time of ${t_a}^P$ must be earlier than that of ${t_b}^B$.

*Property 6.* If $U({t_a}^P) = U({t_b}^B)$, then ${t_a}^B$ cannot overlap with ${t_b}^P$.

As depicted in Figure 6, we give proof by contradiction. Suppose that UAV $u_j$ undertakes task ${t_a}^P$ and task ${t_b}^B$ and task ${t_a}^B$ overlaps with task ${t_b}^P$ on UAV $u_k$. Consider a situation that $u_j$ encounters a damaging problem when executing

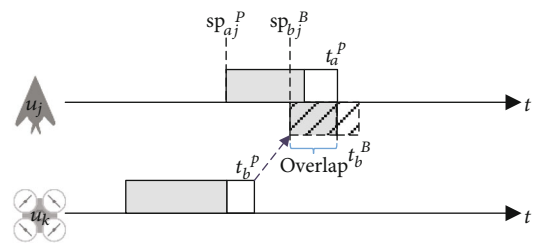


Figure 5: The relationship between start preparation times of overlapping tasks.

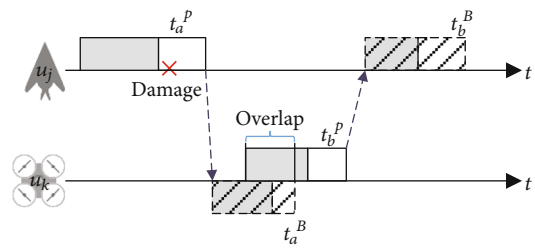


Figure 6: Illustration example for Property 6.

task ${t_a}^P$, task ${t_a}^P$ would fail, and task ${t_a}^B$ would have to start, making primary task ${t_b}^P$ fail because of overlapping. Moreover, task ${t_b}^B$ would also fail owing to $u_j$'s damaging. Consequently, task $t_b$ fails, and the assignment plan is not able to deal with the damage problem of $u_j$.

*Property 7.* If $U({t_a}^P) = U({t_b}^P)$, then ${t_a}^B$ cannot overlap with ${t_b}^B$.

That is to say, when the primary copies of two tasks are assigned to the same UAV, then the backup copies of them cannot overlap. The property can be illustrated in Figure 7.

As depicted in Figure 7, $U({t_a}^P) = U({t_b}^P)$, and ${t_a}^B$ overlaps with ${t_b}^B$. When the UAV $u_j$ is damaged, both ${t_a}^P$ and ${t_b}^P$ would fail, leading to the condition that both ${t_a}^B$ and ${t_b}^B$ need to be executed. However, ${t_a}^B$ and ${t_b}^B$ overlapped in the same UAV; a time conflict happens. Therefore, ${t_a}^B$ cannot overlap with ${t_b}^B$ in the situation that $U({t_a}^P) = U({t_b}^P)$.

It should be noted that each of the seven attributes must not be violated at any time; otherwise, the overlap mechanism may not be able to effectively deal with the damaging problem, thereby affecting the success execution of the tasks.

## 5. Damage-Tolerant Task Assigning Algorithm

Based on the backup mechanism and overlapping models, we present a novel damage-tolerant task assignment algorithm—DTTA—for aperiodic and independent tasks on UAVs in confrontational environments. Specifically, DTTA judiciously considers scheduling objectives so long as damage tolerance. In this section, the primary copy and backup copy assigning algorithm are to be present.

*5.1. Assigning Algorithm for Primary Copies.* As it is easy to be known, the primary copies need to be finished as early as possible for the sake of UAV utilization and, if it has backup

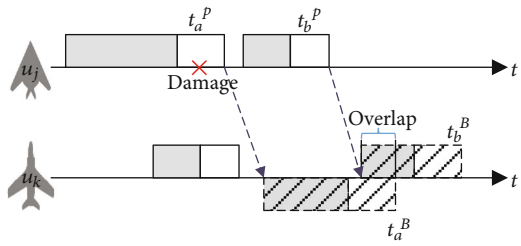


Figure 7: Illustration example for Property 7.

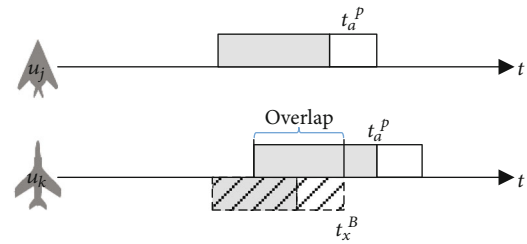


Figure 8: Assigning modes for primary copies.

copies, leaving enough time for its backup copies. Besides, primary copies should be distributed into several UAVs according to Property 7, which would make the overlapping mechanism more feasible. Consequently, the primary copy assignment should work towards the two objectives:

(i) Earliest finish time

(ii) Even load to several UAVs

In addition, the assigning approach should be illustrated. As depicted in Figure 8, for each primary copy, it can be assigned in two kinds of modes. Like task $t_a^P$, the primary copy can be assigned to a UAV, with no overlapping. Moreover, the primary copy can be assigned to a UAV and overlaps with another backup copy like $t_b^P$, if $t_b$ has a backup copy. As the overlapping ways may promote the utilization of UAVs, and leaving more resources for the latter tasks, the second way is used preferentially.

In Algorithm 1, we present the primary copy assigning algorithm in DTTA. Algorithm 1 firstly considers the distance between the task and UAVs and chooses the nearest $\alpha$% for assignment (see lines 1-6). For each UAV in the candidate set, Algorithm 1 would calculate the task's earliest finish time in an overlapping mode on the premise that task $t_i^P$ can overlap UAV $u_j$'s last task. The overlapping qualification is decided by the fact that the environment-threatened value e $t_i$ is larger than the threshold, and $u_j$'s last task is a backup copy with the condition that backup copy was not overlapped (see lines 8-15). In order to obtain a load balance for UAVs, the earliest finish time in a nonoverlapping mode would be calculated for comparison, and the algorithm finally acquires an earlier finish time (see lines 16-21). In line 22, if the algorithm has found a proper UAV $u$ and the corresponding backup copy $t_i^B$ can be scheduled based on $t_i^P$'s earliest finish time, the assigning in the overlapping mode would happen. It would set *AllocateTag* to be true, the task and its backup copy would be allocated, and Algorithm 1 terminates for another task (see lines 23-27). If the overlapping assignment attempt fails, it tries the nonoverlapping mode (see line 28). If the nonoverlapping assignment attempt succeeds, task $t_i^P$ would be scheduled in the nonoverlapping mode (see lines 29-30), and the backup copy of $t_i^P$ would be assigned according to whether it is needed (see lines 31-33); then Algorithm 1 terminates for another task (see line 34). If the task fails to be assigned in both the overlapping mode and nonoverlapping mode, the candidate UAV set would be updated to be the next top $\alpha$% nearest UAVs (see line 35), and a new round assigning attempt would start. If the task fails to be assigned after all UAVs in $U*$ have been scanned, the task is to be rejected (see lines 36-37), and Algorithm 1 terminates for another task. It is worth noting that a task can be assigned in an overlapping mode only when its backup copy can be assigned either (see line 22). The reason is that a task must have a backup copy when it overlaps with other tasks. When in the nonoverlapping assigning mode, a task would try to assign its backup if needed, and the failure of the backup assigning would not lead to the failure of the primary task.

*5.2. Assigning Algorithm for Backup Copies.* For the backup copies, the assignment preference is different. As the damage-tolerant mechanism illustrated before, an earlier finish time of task $t_i^P$ would promote the success probability of its corresponding backup copy $t_i^B$, but the backup copy does not have such preference. Consequently, the assignment of backup copy mainly considers the distance and load balance among UAVs rather than the earliest finish time.

Similar to the case of Figure 8, the assigning for backup copies also has two modes, which are the overlapping mode and non-overlapping mode. And the algorithm is given in Algorithm 2.

From the code, we can see that lines 1-6 are the same as those in Algorithm 1, and the explanation is similar. Line 8 is to make sure all UAVs would be scanned before the task is being rejected. In line 9, we sort the candidate UAV by its unfinished tasks; the reason is that the tasks should be assigned to UAVs as even as possible for higher UAV utilization. Lines 10-17 mean that the task would be assigned in the overlapping mode firstly. If the task fails to overlap with other tasks, the nonoverlapping mode is to be used for assignment (see lines 18-25). Lines 26-27 mean that if the allocation has been succeeded, it would end the scanning loop. Line 28 is used for update candidate UAV set on the condition that existing candidate UAVs cannot satisfy the demands of task $t_i^B$. When all UAVs have been scanned and the task was not assigned, $t_i^B$ would be rejected.

It is worth noting that, line 22 in Algorithm 1 would make sure that the backup copy is to be successfully assigned. But the backup copy of $t_i^P$ in line 32 of Algorithm 1 may be rejected in some cases, decided by the UAVs' status. The reason for this setting is that the backup copy is a prerequisite of overlapping assignment, and but the backup copies are just an optimal goal rather than prerequisites in the nonoverlapping mode.

## 6. Performance Evaluation

In this section, we conduct simulation experiments to verify the effectiveness of the proposed algorithm—DTTA. We

```
Get p_i of arrival task t_i^P;
Let U* ⟵ UAVs in U that satisfies task t_i' demands;
foreach u_j in U* do.
   Calculate the distance d_ij between task t_i and UAV u_j;
Sort U* in an increasing order by the value of d_ij;
U_candidate ⟵ top α% UAV in U*;
AllocateTag ⟵ FALSE; FT ⟵ +∞; u ⟵ NULL;
while !all u_j∈U * has been scanned do.
   OverlapTag ⟵ TRUE;
   foreach u_j in U_candidate do.
      if u_j's assigned last task can be overlapped then.
         Calculate EFT_j(t_i^P) in overlapping mode;
         if EFT_j(t_i^P)<FT & EFT_j(t_i^P) <td_i then.
            FT ⟵ EFT_j(t_i^P);
            u ⟵ u_j;
   foreach u_j in U_candidate do.
      Calculate EFT_j(t_i^P) in non-overlapping mode;
      if EFT_j(t_i^P)<FT & EFT_j(t_i^P) <td_i then.
         OverlapTag ⟵ FALSE;
         FT ⟵ EFT_j(t_i^P);
         u ⟵ u_j;
   if u is not null & t_i^B can be scheduled & OverlapTag = TRUE then.
      AllocateTag ⟵ TRUE;
      Assign t_i^P on UAV u in overlapping mode;
      generate backup copy t_i^B based on FT;
      schedule t_i^B;
      break;
   if u is not null then.
      Assign t_i^P on UAV u in non-overlapping mode;
      AllocateTag ⟵ TRUE;
      if e_ti > et_threshold then.
         generate t_i^B based on FT;
         schedule t_i^B;
      break;
   U_candidate ⟵ next top α% UAV in U*;
   if allocateTag ⟵ FALSE then.
   Reject t_i^P;
```

Algorithm 1: Primary Copies Assigning in DTTA.

compare it with three baseline algorithms: Non-BPoverlapping DTTA (NBPDTTA), Non-BBoverlapping DTTA (NBBDTTA), and No-Backup and No-Overlapping Task assignment Algorithm (NBNOTA).

NBBDTTA is an algorithm derived from DTTA by removing BP overlapping mechanism and NBBDTTA a variant of DTTA by removing BB overlapping mechanism. The difference between NBNOA and DTTA is that there are no backup and overlapping mechanism in DTTA.

The performance metrics, by which we evaluate the effectiveness of algorithms, include:

(i) TGR (task guarantee ratio) is defined to be the percentage of tasks that succeed among all arriving tasks. TGR is used to evaluate the objective in equation (3)

(ii) ETR (execution time ratio) is defined to be the percentage of task execution time among total flying time of UAVs. ETR is used to evaluate the objective in equation (4)

(iii) DR (damaging ratio) is defined to be the percentage of damaged UAVs among all UAVs

*6.1. Simulation Setup.* In order to ensure the repeatability of the experiments, we choose simulation to verify the effectiveness of the proposed algorithms. The detailed setting and parameters are given as follows:

(i) The initial number of UAVs is set to be 300

(ii) The mission area is a rectangle of 3000 m ∗ 2000 m. The UAVs take off from the base at the location (0, 0), and the survival UAVs must return to the base at last

(iii) Each UAV has one or two kinds of abilities, which are chosen from set {*reconnaissance ability*, *striking ability*, *damage assessment ability*}. The initial striking resource is set to be 5, which means a UAV can undertake mostly 5 striking tasks

```
Get p_i of arrival task t_i^B;
Let U* ⟵ UAVs in U that satisfies task t_i' demands;
foreach u_j in U* do.
   Calculate the distance d_ij between task t_i and UAV u_j;
Sort U* in an increasing order by the value of d_ij;
U_candidate ⟵ top α% UAV in U*;
AllocateTag ⟵ FALSE;
while !all u_j∈U* has been scanned do.
   Sort U_candidate by task count in ascending order;
   foreach u_j in U_candidate do.
      if t_i^B can overlap with u_j' last task then.
         Calculate EFT_j(t_i^B) in overlapping mode;
         if EFT_j(t_i^B)<td_i then.
            AllocateTag ⟵ TRUE;
            Assign t_i^B on UAV u_j in overlapping mode;
            st(t_i) ⟵1;
            break;
   if AllocateTag = FALSE;
      foreach u_j in U_candidate do.
      Calculate EFT_j(t_i^B) in non-overlapping mode;
      ifEFT_j(t_i^B)<td_i then.
         AllocateTag ⟵ TRUE;
         Assign t_i^B on UAV u_j in non-overlapping mode;
         st(t_i) ⟵1;
         break;
   if AllocateTag = TRUE;
      break;
   U_candidate ⟵ next top α% UAV in U*;
   if allocateTag ⟵ FALSE then.
   Reject t_i^B;
```

Algorithm 2: Backup Copies Assigning in DTTA.

(iv) The maximum velocity of UAVs is set to be 80 m/s according to current technologies [34]

(v) The tasks arrive at the task assignment system dynamically, following Poisson distribution with the average internal time $1/\lambda$

(vi) The execution time follows a uniform distribution pattern, and the value of execution time is determined by their types. For reconnaissance tasks, the value of en is between [15, 25] and striking tasks en~[25, 35] assessment tasks en~[35, 45]

(vii) We use parameter BaseDeadline to set the deadline of task $t_i$; the formula is as follows:

$$\mathrm{td}_i = \mathrm{ta}_i + \mathrm{BaseDeadline}, \tag{8}$$

where parameters $\mathrm{td}_i$ and $\mathrm{ta}_i$ denote the deadline time and arrival time of $t_i$, and BaseDeadline follows a uniform distribution in the interval $[30, 30 + a * 30]$

(viii) Value $\alpha$ is set to be 20 when assigning primary and backup copies of arrival tasks

(ix) Value $\mu$ is set to be 0.2 when calculating the damage probability of arrival tasks, which means the maximum damaging probability of a task is 20%

The experiments mainly contain four groups, analyzing the effectiveness of backup and overlapping mechanism, and DTTA's performance variation trend with the parameters of $\lambda$, $a$, $\mathrm{et}_{\mathrm{threshold}}$, among which $a$ can be found above. And the intervals of these parameters are listed in Table 2.

*6.2. Performance with the Time Variation.* In this experiment, we study the performance of four algorithms when experiment time varies. It should be noted that the time variation mainly influences the number of UAVs, owing to a damaging problem. In this group of experiments, the tasks are assigned in four kinds of schemas, which are determined according to DTTA, NBPDTTA, NBBDTTA, and NBNOTA, respectively. Figure 9 shows the performance of the four algorithms in terms of *TGR*, *ETR*, and *DR*.

It can be observed from Figure 9(a) that the damaging ratio of all four algorithms increases rapidly with time. At the moment of 100 s, the damaging ratio reaches 15%. At the moment of 800 s, the damaging ratio reaches 78% for an average under the condition that the maximum damaging probability is 20% when executing a task. That means nearly 240 UAVs of initial 300 UAVs have damage in their task

Table 2: Parameters for experiment studies.

| Parameter | Value (fixed)-(varied) |
|---|---|
| ExperimentTime | (600)-(100,200, ⋯, 800) |
| $\lambda$ | (3)-(1, 2, 3, 4, 5, 6, 7) |
| $a$ | (4)-(1, 2, 3, 4, 5, 6, 7) |
| $\mathrm{et}_{\mathrm{threshold}}$ | (0.5)-(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8) |

execution when time is 800 s. The reason can be concluded that the damaging probability increases exponentially with the number of tasks undertaken. As a result, the experiments show that the damaging problem cannot be ignored on the battlefield.

From Figure 9(b), we can observe that DTTA and its deriving algorithms achieve higher performance than NBNOTA for nearly 12%. We can attribute this to the fact that the damage-tolerant mechanism has made its contribution. Besides, we can see that the TGR drops slowly before 500 s and decreases rapidly after 500 s. The reason is that survival UAVs would become less and less with the time. Before 500 s, the survival UAVs can satisfy the tasks' demand, so the TGR only drops slowly. However, the survival UAVs are not enough for arriving tasks after 500 s, so the TGR would decrease quickly. Another phenomenon that can be observed from Figure 9(b) is that the TGR of NBBDTTA drops faster than the TGR of DTTA and NBPDTTA, which means that the BB overlapping model would achieve higher performance than the PB model in the case of lacking UAV resource. The reason is that, when the UAVs are working in heavy burden, the UAVs may encounter much more primary task cancel owing to UAV damage problem in the BP mode, the chain reaction of cancel would lead to resource waste. But in the BB model, only backup copies are overlapping, and no such chain reaction would affect the performance of DTTA and NBPDTTA. We can also find that the NBNOTA would reach higher performance than others after 800 s; this is because backup mechanism and overlapping mode are not suitable for full load environment. That can be explained as follows: the backup mechanism would definitely, although not much, bring the extra expense of resources to guarantee task success probability. When UAVs are in full load, it would be better for UAVs to reject a backup and accept a new task for the reason of no extra resource consumption. So we can make a conclusion that DTTA is not fit for full load UAVs.

We can observe from Figure 9(c) that the difference between the ETRs of four algorithms is very small. This means the proposed algorithm can achieve a higher task success ratio without damaging the UAVs' efficiency. It can be seen in Figure 9(c) that the ETR would grow slowly with time. This is due to the fact that there are fewer UAVs that survive, leading to the fact that each UAV should undertake more tasks, so the ETR will be higher. And the reason for it growing slowly is that a certain part of time is used for transition between tasks' locations. In addition, the ETR of NBBDTTA is smaller than that of DTTA and NBPDTTA. The reason is like the explanation of NBBDTTA dropping faster in the part of Figure 9(b). As for the value of the ETR, it cannot reach nearly 100%; the reason is that flying between different task locations would occupy the usage of time of UAVs.

We can conclude from Figure 9 that the proposed algorithm is effective in enhancing the performance of UAVs to complete tasks in confrontational environments, with no damaging to UAVs' efficiency. And DTTA can perform better among UAVs which are not in full load states. Besides, BB overlapping would achieve higher performance than BP overlapping mode in heavy load conditions.

*6.3. Performance Impact of Arrival Rate.* In this group of experiments, we focus on the impact of arrival rate on performance among DTTA, NBBDTTA, and NBPDTTA. Parameter $\lambda$ varies from 1 to 7 with an increment of 1. The experiment results can be seen in Figure 10.

It can be seen from Figure 10(a) that the damage ratio grows with the increase of $\lambda$. The reason is that, when the value of $\lambda$ is becoming larger, more tasks would arrive in a certain time slot, leading to the condition that each UAV needs to execute more tasks. Therefore, the damage risk would increase, and the DR grows.

The results in Figure 10(b) demonstrate that when the value of $\lambda$ increases, the TGR of all algorithms would decrease slowly before $\lambda = 3$ and drop rapidly after that. We explain this phenomenon as follows. When before $\lambda = 3$, the UAVs have not reached full load, and they can accept some tasks for execution, so the TGR only decreases slowly. When after $\lambda = 3$, many more tasks would come in a certain time slot, and existing UAV resources would be much less than the demands, leading to the fact that UAVs have no choice but to reject the majority of arriving tasks, so the TGR drops rapidly. An interesting observation is that the performance of DTTA, NBBDTTA, and NBPDTTA is higher than that of NBNOTA before $\lambda = 3$, and NBNOTA would perform better after $\lambda = 3$. This is due to the fact that DTTA and its deriving algorithms are not fit for heavy load conditions, which is consistent with the explanation in Figure 9(b). Besides, we can see that the proposed DTTA performs better than NBBDTTA and NBPDTTA when the value of $\lambda = 3$ varies from 1 to 4. Therefore, when in suitable conditions, the proposed DTTA is definitely effective.

Consequently, we can conclude from Figure 10 that parameter $\lambda$ mainly affects the load state for UAV swarms. And when not facing a full load environment, the DTTA can perform stably better than other algorithms with the variation of $\lambda$.

*6.4. Performance Impact of Deadline.* We investigate the impact of the task's deadline on the performance of the DTTA. Parameter $a$, which can be found in equation (8), varies from 1 to 7 with an increment of 1.

Figure 11 shows that when the deadline is very short, the DTTA, NBBDTTA, and NBPDTTA achieve very similar performance with NBNOTA. We can attribute the reason to the fact that tasks would fail to schedule backup copies owing to a short deadline. When the deadline becomes longer, the performance of all algorithms increases rapidly, but the DTTA grows faster. This is because the backup mechanism makes

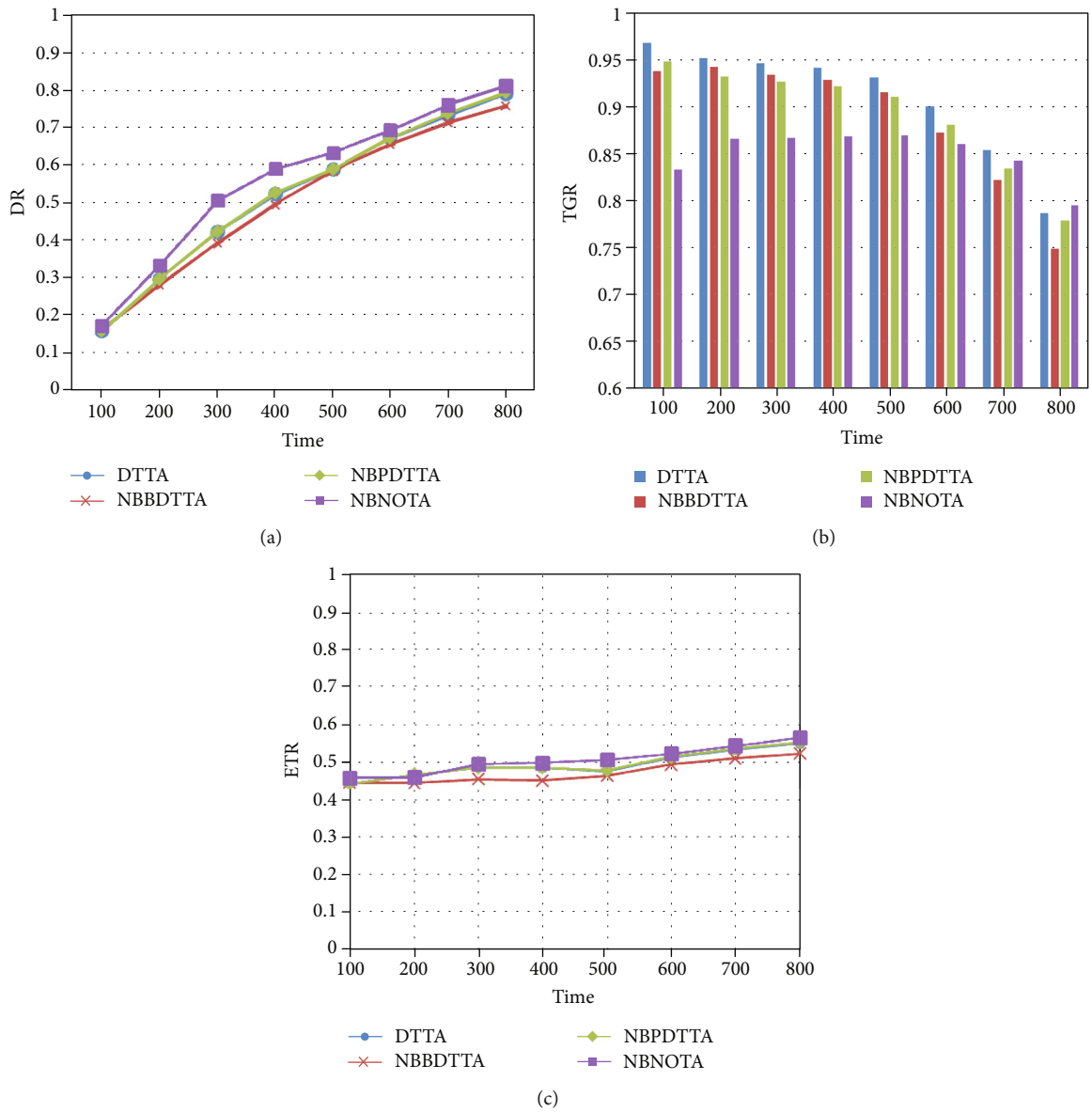


Figure 9: Performance impact of time.

its contribution. And the DTTA can perform better than NBNOTA for nearly 7% when the value of *a* reaches 4. However, with the increment of *a* after the value of 4, the performance would be kept the same. The reason is that when the backup mechanism has reached its potential, a longer deadline is meaningless for performance promotion.

We can conclude from Figure 11 that the deadline would affect the performance of DTTA only when the deadline is short enough to influence the success of assigning backup copies.

*6.5. Performance Impact of Backup Threshold.* In this group of experiments, the impact of the threshold value for backup or not would be studied. The results are presented in Figure 12.

It can be observed that the performance of DTTA, NBPDTTA, and NBBDTTA would change with the variation of $et_{threshold}$. However, the performance of NBNOTA was stable. The reason is that NBNOTA does not apply backup mechanism, and the value of $et_{threshold}$ would not affect the scheduling of tasks under NBNOTA. In addition, DTTA would achieve the highest performance when the value of e $t_{threshold}$ is 0.5. We can attribute this to that, when the e $t_{threshold}$ is smaller than the proper value, too many backups would bring extra resource expense and, when the $et_{threshold}$ is bigger than the proper value, too little backup would make the overlapping mechanism and backup mechanism cannot reach their potential. Obviously, the proper value of e $t_{threshold}$ is 0.5 under the experiment settings of this group.

\Another phenomenon found in Figure 12 is that DTTA and its deriving algorithms would keep better performance than NBNOTA. This result proves that the damage-tolerant algorithm would enhance the task success ratio. We can also find that the DTTA achieves the highest performance among the three damage-tolerant algorithms. It means that the BB

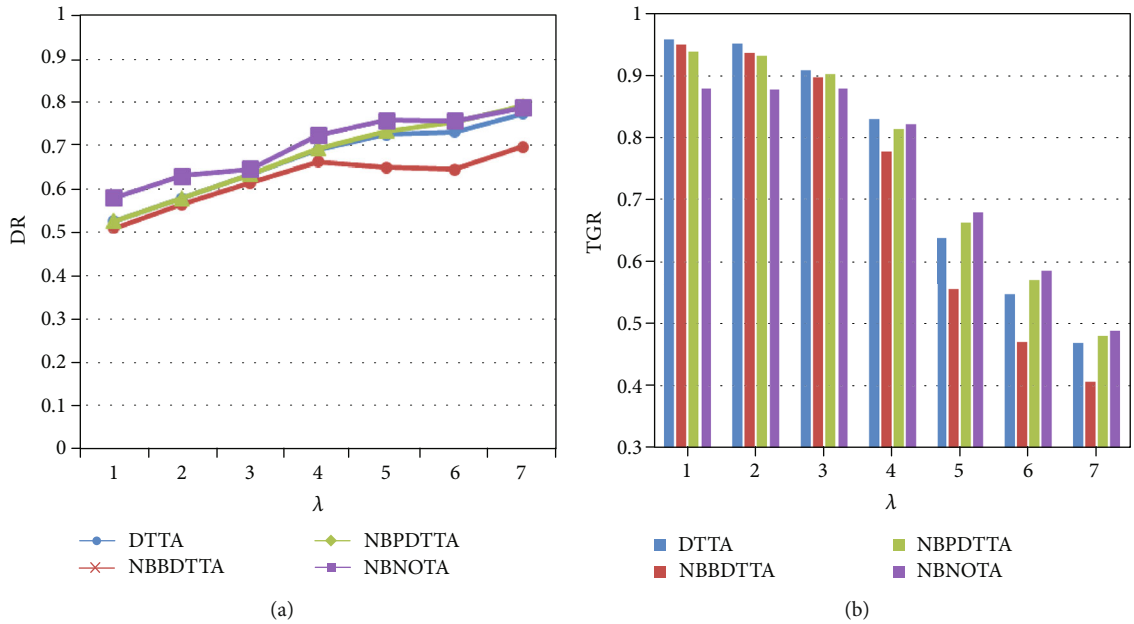


Figure 10: Performance impact of $\lambda$.

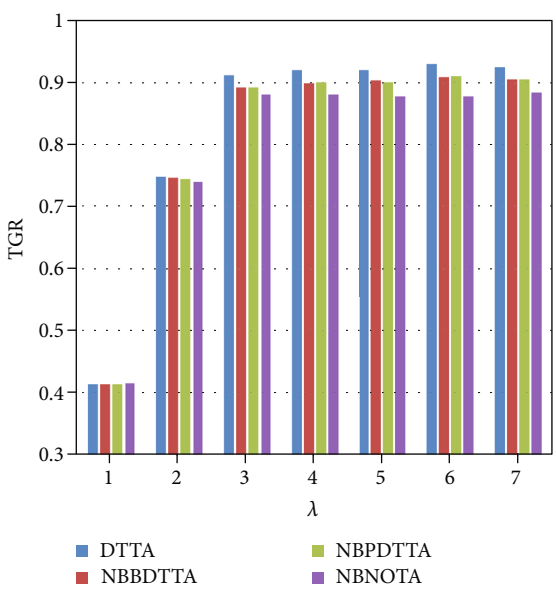


Figure 11: Performance impact of $a$.

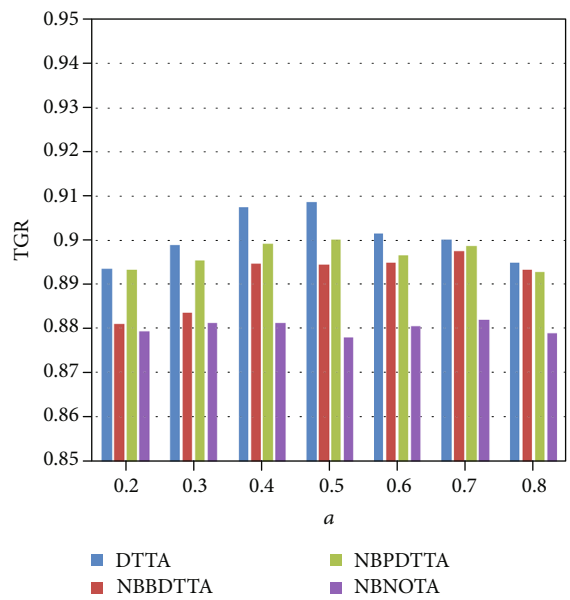


Figure 12: Performance impact of $\mathrm{et}_{\mathrm{threshold}}$.

overlapping model and PB overlapping model should work together for TGR performance in any value of $\mathrm{et}_{\mathrm{threshold}}$. In addition, by comparing the performance of $\mathrm{et}_{\mathrm{threshold}} = 0.5$ in Figure 12, $a = 4$ in Figure 11, and $\lambda = 3$ in Figure 10, we can see the difference among them is very small. That identifies the stability of the proposed algorithm DTTA.

## 7. Conclusions and Future Work

In this study, we investigated the damage-tolerant assigning problem for UAV swarms in confrontational environments and proposed a novel damage-tolerant task assignment algorithm named DTTA. The DTTA employs backup mechanism to enhance the probability of task success and applies the BB and BP overlapping models to promote the utilization of UAVs. Based on the models, we investigate the overlapping properties to avoid collision between tasks. What is more, a no backup algorithm NBNOTA and two algorithms, named NBBDTTA and NBPDTTA, derived from DTTA are proposed to be the baseline. We conducted extensive simulation studies by using random synthetic tasks, indicating that DTTA is a feasible task allocation algorithm in confrontational environments. When not in full load environment, the DTTA can enhance the task success ratio for about 10% or more (see the performance of time 200 s in Figure 9(b)), without damaging the efficiency of UAV utilization.

Our DTTA is the first of its kind reported in the literature, because DTTA takes the assigning algorithm, damage-tolerant mechanism, and overlapping model into collaborative thinking. And the experiments verified its advantages. In our future work, we will enhance the task and UAV models to be more in line with the real case. Moreover, we will apply the algorithm into real systems to deal with real tasks for damage tolerance.

## Data Availability

The experiment data can be reached by contacting the corresponding author (wdbao@nudt.edu.cn).

## Conflicts of Interest

The authors declare that there are no conflicts of interest about the publication of this paper.

## Acknowledgments

This work is supported by the National Science Foundation of China (No. 71702186), the Scientific Research Project of National University of Defense Technology (Grant No. ZK19-03), the Military Graduate Research Project (No. 1204054419068KY), and Postgraduate Research Innovation Project in Hunan Province (Grant No. CX2018B021).

## References

[1] J. Chen, H. Liu, J. Zheng et al., "Damage degree evaluation of earthquake area using UAV aerial image," *International Journal of Aerospace Engineering*, vol. 2016, Article ID 2052603, 10 pages, 2016.

[2] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[3] L. Huang, X. Yu, and X. Zuo, "Edge detection in UAV remote sensing images using the method integrating Zernike moments with clustering algorithms," *International Journal of Aerospace Engineering*, vol. 2017, Article ID 1793212, 7 pages, 2017.

[4] B. D. Song, K. Park, and J. Kim, "Persistent UAV delivery logistics: MILP formulation and efficient heuristic," *Computers & Industrial Engineering*, vol. 120, pp. 418–428, 2018.

[5] K. Kuma, C. S. Reddy, and J. Shanmugam, "Design and simulation of blending function for landing phase of a UAV," *Defence Science Journal*, vol. 58, no. 3, pp. 315–326, 2008.

[6] K. Giles and K. Giammarco, "Mission-based architecture for swarm composability (MASC)," *Procedia Computer Science*, vol. 114, pp. 57–64, 2017.

[7] X. Hu, H. Ma, Q. Ye, and H. Luo, "Hierarchical method of task assignment for multiple cooperating UAV teams," *Journal of Systems Engineering and Electronics*, vol. 26, no. 5, pp. 1000–1009, 2015.

[8] F. Xiong, A. Li, and H. Wang, "An SDN-MQTT based communication system for battlefield UAV swarms," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 41–47, 2019.

[9] D. H. Kim, K. Lee, and M. Y. Park, "UAV-based localization scheme for battlefield environments," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, pp. 562–567, San Diego, CA, USA, 2013.

[10] A. Zhang, D. Zhou, P. Yang, and M. Yang, "Event-triggered finite-time consensus with fully continuous communication free for second-order multi-agent systems," *International Journal of Control Automation & Systems*, vol. 17, no. 4, pp. 836–846, 2019.

[11] S. Ghosh, R. Melhem, and D. Mosse, "Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 3, pp. 272–284, 1997.

[12] C. Shumacher, P. Chandler, M. Pachter, and L. Pachter, "UAV task assignment with timing constraints via mixed-integer linear programming," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, p. 6410, Chicago, IL, USA, 2004.

[13] C. Liu and A. Kroll, "Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems," *Springer Plus*, vol. 5, no. 1, article 1361, 2016.

[14] H. R. Boveiri, "An incremental ant colony optimization based approach to task assignment to processors for multiprocessor scheduling," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 4, pp. 498–510, 2017.

[15] T. Shima, S. J. Rasmussen, A. G. Sparks, and K. M. Passino, "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms," *Computers & Operations Research*, vol. 33, no. 11, pp. 3252–3269, 2006.

[16] H.-S. Wu and F.-M. Zhang, "Wolf pack algorithm for unconstrained global optimization," *Mathematical Problems in Engineering*, vol. 2014, Article ID 465082, 17 pages, 2014.

[17] M.-H. Kim, B. Hyeoncheol, and L. Seokcheon, "Resource welfare based task allocation for UAV team with resource constraints," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3-4, pp. 611–627, 2015.

[18] L. Thomas, A. Rachid, and L. Simon, "A distributed tasks allocation scheme in multi-UAV context," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pp. 3622–3627, New Orleans, LA, USA, 2004.

[19] L. Shen, C. Jing, and W. Nan, "Overview of air vehicle mission planning techniques," *Acta Aeronautica et Astronautica Sinica*, vol. 35, no. 3, pp. 593–606, 2015.

[20] Q. Cheng, D. Yin, J. Yang, and L. Shen, "An auction-based multiple constraints task allocation algorithm for multi-UAV system," in *2016 International Conference on Cybernetics, Robotics and Control (CRC)*, pp. 1–5, Hong Kong, China, 2004.

[21] H. Wu, H. Li, R. Xiao, and J. Liu, "Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm," *Physica A: Statistical Mechanics and its Applications*, vol. 491, pp. 127–141, 2018.

[22] F. Bateman, N. Hassan, and O. Mustapha, "Fault diagnosis and fault-tolerant control strategy for the aerosonde UAV," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 2119–2137, 2011.

[23] S. Park, J. Bae, Y. Kim, and S. Kim, "Fault tolerant flight control system for the tilt-rotor UAV," *Journal of the Franklin Institute*, vol. 350, no. 9, pp. 2535–2559, 2013.

[24] B. Yu, Y. Zhang, J. Yan, Y. Qu, and Z. Liu, "Fault-tolerant control with linear quadratic and model predictive control techniques against actuator faults in a quadrotor UAV," in *2013*

*Conference on Control and Fault-Tolerant Systems (SysTol)*, pp. 6294–6299, Xian, China, 2013.

[25] T. Peni, B. Vanek, Z. Szabo, and J. Bokor, "Supervisory fault tolerant control of the GTM UAV using LPV methods," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 1, pp. 117–131, 2015.

[26] G. Manimaran and C. S. R. Murthy, "A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 11, pp. 1137–1152, 1998.

[27] X. Qin and J. Hong, "A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems," *Parallel Computing*, vol. 32, no. 5-6, pp. 331–356, 2006.

[28] J. Wang, W. Bao, X. Zhu, L. T. Yang, and Y. Xiang, "FESTAL: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2545–2558, 2015.

[29] Q. Zheng, B. Veeravalli, and C. K. Tham, "On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 380–393, 2009.

[30] H.-X. Chen, Y. Nan, and Y. Yang, "Multi-UAV reconnaissance task assignment for heterogeneous targets based on modified symbiotic organisms search algorithm," *Sensors*, vol. 19, no. 3, p. 734, 2019.

[31] Z. Zhen, D. Xing, and C. Gao, "Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm," *Aerospace Science and Technology*, vol. 76, pp. 402–411, 2018.

[32] J. F. Galarreta, N. Kerle, and M. Gerke, "UAV-based urban structural damage assessment using object-based image analysis and semantic reasoning," *Natural Hazards and Earth System Sciences*, vol. 15, no. 6, pp. 1087–1101, 2015.

[33] X. Zhu, J. Wang, X. Qin, J. Wang, Z. Liu, and E. Demeulemeester, "Fault-Tolerant scheduling for real-time tasks on multiple earth-observation satellites," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3012–3026, 2015.

[34] "The 10 longest range unmanned aerial vehicles (UAVs)," February 2020, https://www.airforce-technology.com/features/featurethe-top-10-longest-range-unmanned-aerial-vehicles-uavs/.