drones

Article

A Distributed Task Scheduling Method Based on Conflict Prediction for Ad Hoc UAV Swarms

Jie Li * and Runfeng Chen

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
* Correspondence: lijie09@nudt.edu.cn

Abstract: UAV swarms have attracted great attention, and are expected to be used in scenarios, such as search and rescue, that require many urgent jobs to be completed in a minimum time by multiple vehicles. For complex missions with tight constraints, careful assigning tasks is inseparable from the scheduling of these tasks, and multi-task distributed scheduling (MTDS) is required. The Performance Impact (PI) algorithm is an excellent solution for MTDS, but it suffers from the suboptimal solution caused by the heuristics for local task selection, and the deadlock problem that it may fall into an infinite cycle of exchanging the same task. In this paper, we improve the PI algorithm by integrating a new task-removal strategy and a conflict prediction mechanism into the task-removal phase and the task-inclusion phase, respectively. Specifically, the task-removal strategy results in better exploration of the inclusion of more tasks than the original PI by freeing up more space in the local scheduler, improving the suboptimal solution caused by the heuristics for local task selection, as done in PI. In addition, we design a conflict prediction mechanism that simulates adjacent vehicles performing inclusion operations as the criteria for local task inclusion. Therefore, it can reduce the deadlock ratio and iteration times of the MTDS algorithm. Furthermore, by combining the protocol stack with the physical transmission model, an ad-hoc network simulation platform is constructed, which is closer to the real-world network, and serves as the supporting environment for testing the MTDS algorithms. Based on the constructed ad-hoc network simulation platform, we demonstrate the advantage of the proposed algorithm over the original PI algorithm through Monte Carlo simulation of search and rescue tasks. The results show that the proposed algorithm can reduce the average time cost, increase the total allocation number under most random distributions of vehicles-tasks, and significantly reduce the deadlock ratio and the number of iteration rounds.

Keywords: UAV swarms; distributed scheduling; mobile ad hoc networks

check for updates

Citation: Li, J.; Chen, R. A Distributed Task Scheduling Method Based on Conflict Prediction for Ad Hoc UAV Swarms. *Drones* **2022**, *6*, 356. https://doi.org/10.3390/drones6110356

Academic Editors: Diego González-Aguilera and Vincent A. Cicirello

Received: 1 October 2022
Accepted: 11 November 2022
Published: 15 November 2022

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

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

1. Introduction

In recent years, UAV swarms have attracted great attention due to their low cost, decentralized layout, and a certain degree of scale, and are expected to be used in post-disaster search and rescue, pollution monitoring and traceability, cargo sorting and delivery, and other fields. In contrast to the simple bionic clustering behaviors such as migration, foraging, nesting, hunting, and resistance achieved by the stigmergy-based swarm algorithms, the complex missions in the above real-world application need the UAV swarms work together to assign tasks and schedule them, which cannot be separated from multi-agent scheduling [1].

Multi-agent scheduling is a classical combinatorial optimization problem [2]; therefore, obtaining the global optimal solution is computationally intensive [3]. Earlier studies used a centralized approach to generate and distribute plans for all vehicles by using a central server capable of gathering system-wide information (situational awareness), either on the ground (the fleet's ground control terminal) or on one of the agents selected as the master [4]. In this approach, the overall objective function of the concern problem can be

explicitly minimized or maximized based on the collection of all vehicle information on a central server. Therefore, their main advantage lies in the ability to optimize the overall objective function (and thus obtain the approximate optimal solution) by some heuristic algorithms, such as genetic algorithm (GA), ant colony algorithm (ACO), particle swarm optimization (PSO) [5], or the metaheuristic approach [6], etc. However, they have several disadvantages for the low-cost swarm [7]: First, the computational requirements are high because the computational operations are placed on a central server for global optimization of the multi-agent scheduling problem; Second, each agent is required to communicate with the central server, and all vehicles need to communicate their situational awareness (SA) to the central server, which will bring a heavy communication burden. Third, centralized approaches are prone to single points of failure [8].

To this end, the swarm can only resort to decentralized approaches [9–13], which could increase the mission range and remove the single points of failure. They are roughly divided into three implementation approaches: redundant central computing, optimization-based approach and market auction approach. Redundant central computing instantiates the centralized scheduler on each vehicle in order to achieve distribution [14–16]. These approaches often assume perfect communication links with unlimited bandwidth, since every vehicle must have the same SA. Inconsistencies in SA can lead to allocation conflicts because each vehicle will perform a centralized optimization using a different information set. Optimization-based approach utilize distributed constraint optimization, game theory, metaheuristics, or other optimization techniques (distributed GA, distributed PSO and so on) to obtain higher optimization performance and computational efficiency [16–18]. However, these methods rely on high frequency and high speed data exchange. Market auction approach is where vehicles bid on tasks in a greedy strategy, the higher bidder wins the task, and the suboptimal solution can be obtained through multiple rounds of bidding [3,19,20]. In these types of algorithms, vehicles bid on tasks with values based solely on their own SA. Even if there are inconsistencies in their SA, they can naturally converge to a conflict-free solution. When UAV swarm perform tasks, the spatial positions of UAVs change rapidly, and slow task scheduling means that the generated schedulers is invalid, which puts forward higher requirements for the timeliness of the algorithm. Compared with the two other approaches, the market auction approach has higher computational efficiency, and lower communication requirements and is more suitable for UAV swarm because of its local greedy strategy and simple consensus rules.

The consensus-based bundle algorithm (CBBA) [19] and the performance impact algorithm (PI) [3] are two of the most representative market auction methods, and many methods in this field are derived from them. Both CBBA and PI essentially take the 'significance' of a task to its assigned agent as the basis for selecting and scheduling tasks, reflecting the value or cost of the task for the global optimization objective. The main difference is that PI uses the native significance as the auction price to achieve better overall optimization performance; The CBBA introduces additional bundling sets and diminishing marginal gain (DMG) pricing mechanisms to ensure that auctions are not deadlocked. It has been shown that CBBA produces the same solution as the centralized sequential greedy procedures, and guarantees 50% optimality, and can ensure that the auction process is deadlock-free, but at the cost of global optimality, because it essentially distorts the native value of the bidding task [21]. In contrast, PI auctions are based on native significance, which achieves better overall optimization, but has more iterations. It also faces frequent deadlock problems (i.e., two or more UAVs occasionally get caught in an infinite cycle of exchanging the same tasks), which can only be broken out of the loop by truncating the auction process, resulting in the blocking of the optimization process. By analyzing the advantages and disadvantages of the two algorithms, this paper considers whether the global optimality of the problem can be further improved, and the bidding rounds and deadlock probability can be reduced, which is of great benefit to UAV swarm task scheduling.

In addition to the algorithm, another issue that cannot be ignored is the communication model on which the algorithm verification is based. Decentralized scheduling methods rely on the communication network to complete negotiation or arbitration to resolve task conflicts. As far as we know, most algorithms assume perfect communication with topologically fully connected, unrestricted bandwidth, and ideal channels. There are also some algorithms tested under different topological connection, such as Zhao et al. [3] use row, star, circular and mesh networks to simulate different communications scenarios to test the PI algorithm. Johnson et al. [22] compare the asynchronous consensus-based bundle algorithm (ACBBA) and CBBA in the fully connected network and row topology; Ismail et al. [23] use a simple disk communication model to compare the decentralized hungarian-based approach (DHBA) and the consensus-based auction algorithm (CBAA); In the past two years, a few scholars have begun to pay attention to the performance differences of decentralized scheduling methods under different channel transmission characteristics such as channel attenuation and packet loss. For example, Nayak et al. [24] compare the performance of CBAA, ACBBA, DHBA, PI, and the hybrid information and plan consensus (HIPC) based on Bernoulli, Gilbert-Elliot and Rayleigh fading communication models; Carrillo et al. [25] use Rayleigh attenuation model to evaluate the current communication level, which is used to select the scheduling algorithm that best fits it. From the above investigation, if the communication factors are considered, the simulation tests of the decentralized scheduling algorithm are based on the transmission characteristics of communication channels, which are mainly divided into two aspects: one is to assume different network topologies and compare the algorithm iteration rounds (the difference caused by neighboring message propagation); the other is to simulate the channel state such as packet loss and evaluate the performance of the algorithm (such as the allocation number, time cost, etc.). Real-world networking communication should not only consider the underlying physical transmission characteristics, but also the working process of the upper-layer communication protocol stack. More reasonable network communication simulation should combine the two, which is one of the work to be considered in this paper.

A distributed task scheduling method with better global optimization performance, fewer iteration rounds, and lower deadlock probability is of great benefit to the real-world applications such as search-and-rescue. This paper modifies the baseline PI algorithm by respectively integrating a new removal strategy and a conflict prediction operator into the tasks removal phase and inclusion phase of the algorithm. The main contributions are as follows.

(1) In the proposed method, a new task removal strategy is adopted to release more space of the scheduler to explore the addition of new tasks, so as to improve the overall optimization. Then, a conflict prediction operator is added to the task inclusion phase of the algorithm to reduce the number of iterations and the probability of deadlock.

(2) By integrating the communication protocol stack of the upper layer network and the physical transmission model of the bottom layer, an ad-hoc network simulation platform is constructed, which is closer to the real world communication, and serves as the supporting environment for the performance testing of decentralized scheduling algorithms.

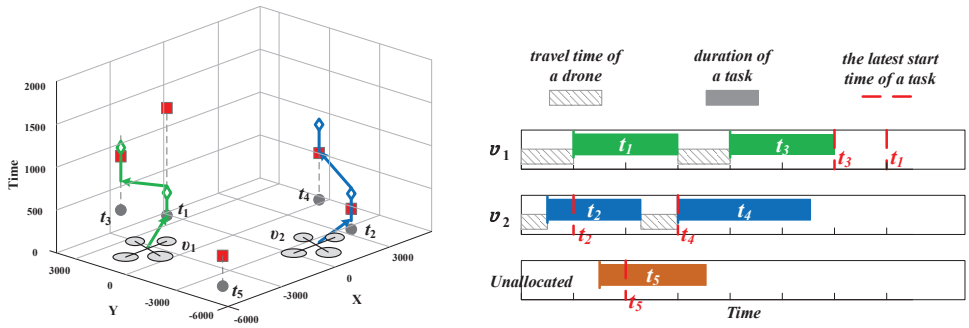
(3) Monte Carlo simulation experiments on the ad-hoc network simulation platform were carried out, and a large number of results show that, compared with PI, the proposed method can reduce the average time cost and increase the total allocation number under most of the random distributions of vehicles and tasks, and significantly reduce the deadlock ratio and the number of iteration rounds.

The rest of this paper is organized as follows. Section 2 briefly describes the distributed task scheduling problem and existing market-based methods. Section 3 presents a distributed task scheduling method based on conflict prediction. Section 4 presents the ad hoc network protocol stack and channel transmission simulation model. Section 5 discusses the performance of the algorithm. Finally, Section 6 concludes the paper.

2. Preliminaries

2.1. Problem Formulation

As shown in Figure 1, consider a variety of heterogeneous UAVs performing a search and rescue mission, the purpose is to cooperate with each other to rescue a number of survivors discovered, who require food supplies or medical provisions [3]. Each survivor must be visited by one UAV before a certain point in order to be deemed rescued, followed by the duration of the rescue. Every UAV always follows a minimum time path to visit all of the unvisited survivors allocated to it, while not required to return to its initial location.



(**a**) (**b**)

Figure 1. Schematic diagram of scenario, where v_1 executes tasks t_1, t_3 in turn to provide food, v_2 executes t_2, t_4 to supply medicine, with remaining task t_5 unallocated. (**a**) the dots represent tasks, dashed lines represent available time periods for tasks, red rectangles represent deadlines for task start, solid lines with arrows represent task schedulers and diamond patterns represent the duration of task execution. (**b**) corresponding task timelines, where the red dotted lines indicate the tasks' latest start time, the diagonal boxes indicate the transition time of the vehicles, the color-filled box indicates the duration of task execution.

The scenarios similarities with the traveling salesman problem (TSP), a well-known NP-hard combinatorial optimization problem. The objectives considered here are comparable to the constraints of two variants of the TSP: the team orienteering problem with time windows (TOPTW) [26] and the K-traveling repairmen problem (K-TRP) [27]. These objectives and constraints are applicable to a variety of scenarios such as those found in target tracking, pick-up and delivery, logistics, and any scenario that requires many urgent jobs to be completed in a minimum time by multiple vehicles. One challenge in using a swarm is to scheduling them to perform tasks while optimizing one or more objectives, for example, to minimise the average waiting time before their rescue, and to maximize the number of rescued survivors.

To formulate the problem mathematically, a set of n heterogeneous UAVs are defined by $\mathbf{V} = [v_1, v_2, \ldots, v_n]$, and a set of m tasks are defined by $\mathbf{T} = [t_1, t_2, \ldots, t_m]$. Each UAV v_i selects the candidate tasks from $\mathbf{T}$ and sequences them to form its scheduler $\mathbf{p}_i$ respecting time constraints. Whether a task is candidate for an UAV depends on two aspects: one is whether the UAV can undertake this type of task, which is limited by the fact that low-cost UAVs usually only have single task capability; another aspect is whether the task is within the limited range of the vehicle. The optimization objective $\mathcal{J}$ of the scheduling problem is to minimize the average waiting time under the premise of allocating as many tasks as possible:

$$\mathcal{J} = \min \frac{1}{m} \sum_{i=1}^{n} \sum_{k=1}^{|\mathbf{p}_i|} c_{i,k}(\mathbf{p}_i) \tag{1}$$

in which the time cost of a task t_k in $\mathbf{p}_i$, defined as $c_{i,k}(\mathbf{p}_i)$, is the predicted waiting time taken by the UAV to arrive at the location of the task t_k. This time includes the duration of earlier tasks in $\mathbf{p}_i$ and travel time to and from those earlier tasks, but does not include

the duration of the execution of t_k. The objective of minimizing waiting time measures the cost of a task scheduling as the time it takes to start serving the task from the start of the drone's schedule, i.e., the total time the survivor must wait before being attended to. $|\mathbf{p}_i|$ is the number of assigned tasks for v_i.

The corresponding constraints are listed below:

$$\begin{cases} |\mathbf{p}_i| \leq N_i, & \forall i \in \{1, \ldots, n\} \\ \mathbf{p}_i \cap \mathbf{p}_j = \emptyset, & \forall i \neq j \in \{1, \ldots, n\} \\ \bigcup\limits_{i}^{z} \mathbf{p}_i \subseteq \mathbf{T}, & \forall i, z \in \{1, \ldots, n\} \\ c_{i,k}(\mathbf{p}_i) \leq s_{\mathbf{p}_{i,K}}, & \forall i \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\} \end{cases} \tag{2}$$

where the first constraint represents the capacity N_i of the vehicle v_i. The second constraint ensures that each task is assigned to at most one vehicle or left unassigned because it is beyond the capabilities of all vehicles. The validity of scheduling is guaranteed by the third constraint where any scheduler's combination is a subset of the task set $\mathbf{T}$. The fourth constraint guarantees that $c_{i,k}(\mathbf{p}_i)$ of vehicle v_i initiating the kth task in its scheduler $\mathbf{p}_i$ should be before the task deadline $s_{\mathbf{p}_{i,K}}$.

2.2. Scheduling with the Market-Based Approach

In general, the market auction approach consists of two phases: the task inclusion phase and the communication and consensus phase. In the first phase, each UAV recursively inserts tasks locally in a greedy strategy until the capacity of the vehicle is reached. In the second phase, the UAV shares its local scheduler with neighbors and resolves allocation conflicts based on a consensus rule. The two phases alternate repeatedly until no further changes are made to the schedules of all vehicles.

During the task inclusion phase, the UAV measures the impact of inserting each candidate task t_q on the cumulative time cost of tasks already in its schedule $\mathbf{p}_i$, called significance, to determine which task should be included in $\mathbf{p}_i$. The specific process for calculating the marginal significance of inserting t_k at each position l in $\mathbf{p}_i$ is as follows:

$$\omega_k^{\oplus}(\mathbf{p}_i, t_k) = \min_{l=1}^{|\mathbf{p}_i|} \sum_{z=l}^{|\mathbf{p}_i|+1} c_{i,z}(\mathbf{p}_i \oplus_l t_k) - \sum_{z=l}^{|\mathbf{p}_i|} c_{i,z}(\mathbf{p}_i) \tag{3}$$

where $\oplus_l$ indicates that the task t_q is inserted into the l_{th} position of $\mathbf{p}_i$. Since the inserted task t_q only affects the start times of its subsequent tasks in $\mathbf{p}_i$, the formula calculates the cumulative time delay of the l_{th} task and the tasks after it. A list to store the marginal significance of each task is kept on each UAV and is defined as $\gamma_i^{\oplus} = \left[w_1^{\oplus}, \ldots, w_m^{\oplus}\right]$ for v_i.

After the marginal significance of all candidate tasks have been computed, v_i selects for inclusion the task whose marginal significance can improve upon task's significance the most. A marginal siginificance of t_k in $\mathbf{p}_i$ lower than the siginificance of t_k's owner v_j indicates that the overall cost can be reduced if t_k is reallocated to v_i. The maximum difference is computed as $\max_{k=1}^{m}\{\gamma_{i,k} - \gamma_{i,k}^{\oplus}\} > 0$, in which the significance $\gamma_{i,k}$ of task t_k in scheduler $\mathbf{p}_i$ is defined as follows.

$$w_k^{\ominus}(\mathbf{p}_i, t_k) = \sum_{z=b}^{|\mathbf{p}_i|} c_{i,z}(\mathbf{p}_i) - \sum_{z=b+1}^{|\mathbf{p}_i|} c_{i,z}(\mathbf{p}_i \ominus t_k) \tag{4}$$

where b is the position of task t_k in v_i's scheduler, and $\mathbf{p}_i \ominus t_k$ denotes $\mathbf{p}_i$ with t_k removed. The significance of a task t_k is referred to formally as $w_k^{\ominus}$ and each UAV stores the vector $\gamma_i = \left[w_1^{\ominus}, \ldots, w_m^{\ominus}\right]$.

During the communication and conflict resolution phase, each UAV shares its schedulers with neighbors and resolve conflicting allocations. As two or more UAVs may be assigned the same task, the consensus procedure introduced in [19] is used to resolve these

conflicting assignments. A lower significance indicates a more optimal schedules, therefore UAVs with a higher significance for a conflicting assignment release the task.

3. Method

3.1. Basic Idea

In contrast, with its superiority in optimization performance, PI is the best benchmark method for solving distributed scheduling problems. However, as mentioned above, it still has room for improvement in optimization, efficiency and frequent deadlocks (i.e., two or more UAVs occasionally get caught in an infinite cycle of exchanging the same tasks). In this section, we modify it to achieve an overall improvement in optimality, an improvement in deadlock problems, and a reduction in the number of iterations. Here, we follow the two-stage algorithm framework of the market auction method. Without loss of generality, we begin with the communication and consensus phase of an intermediate iteration to illustrate the basic idea of the proposed method.

In the communication and consensus phase, PI resolves conflicts by iteratively deleting tasks that failed in the previous round, item by item. Whenever a task is deleted, the algorithm updates the significance of the remaining underbid tasks, causing them to decrease. If the updated significance is lower than the winning price of the task in the previous round, the original task to be deleted will be preserved. In fact, this method of iteratively updating the significance of the remaining failed tasks has two disadvantages: first, the failed tasks in the last round of bidding may still remain in the local scheduler, which makes it difficult to absorb more tasks in the adjacent task inclusion phase, which restricts the increase of the total task allocation; second, the conflicting assignment is not completely resolved. Failed tasks remain in the local scheduler and cannot be removed until later rounds, which potentially increases the number of iterations. This is essentially a trade-off between exploration and utilization [28]. Although PI achieve good optimization with native significance, its conservative removal strategy restrict the exploration ability of the algorithm. To this end, this article eliminates the step of iteratively updating significance and includes freeing up more local scheduler space for subsequent tasks. To this end, this article removes the iterative update importance operation to free up more local scheduler space, which helps to absorb more tasks in the adjacent task inclusion phase.

In the task inclusion phase, PI iteratively includes tasks item by item by comparing the marginal significance of the task to be inserted with the bid price of the task. This method can not only achieve lower overall task cost by exchanging existing tasks among swarms, but also realize the insertion of new tasks to obtain more assigned tasks. Through communication, each UAV obtains the local scheduling tasks of other vehicles in the last iteration cycle (which is also the information input for conflict resolution). Based on this, it can predict the included operations of other UAVs in this cycle and judge whether their included tasks are the same as those of the local vehicle in advance. And calculate their marginal significance to determine whether the task is inserted locally. This strategy is expected to reduce the iteration rounds by resolving conflicts in advance. In addition, PI is faced with deadlock problem, that is, two or more vehicles are often in an infinite loop exchanging the same task. Preliminary experiments of running PI show that it mostly occurs between adjacent auction rounds. The above strategy can also avoid some deadlock problems that occur in adjacent rounds. To this end, this paper alleviates the deadlock problem by mutually predicting the task inclusion operations of the current rounds of the respective UAVs to decide whether a task should be added.

Next, we use the following schematic example to illustrate the above basic idea.

Example 1. *As shown in Figure 2, without loss of generality, it is assumed that two vehicles use a greedy algorithm to select and schedule four tasks of a_1, b_1, c_1, d_1 and b_2, a_2, c_2, d_2 in the previous scheduling cycle, where subscripts 1 and 2 are used to distinguish the vehicles to which they belong. For the task removal phase, after the communication of bids, referring to the consensus rule table, the two vehicles directly remove the tasks with no dominant price and get schedules a_1, c_1 and b_2, c_2*

respectively. For the task inclusion phase, each vehicle predicts each other for one scheduling cycle as the basis for judging and inserting tasks. Vehicle 1 was originally intended to contain the task f, but it is not inserted because it is predicted in advance that vehicle 2 will also contain this task through the predictor, and f_2 is smaller than f_1. To this end, vehicle 1 still tries to insert other tasks e, g on the basis of paths a_1, c_1, and obtains the schedule e_1, a_1, c_1 at the end of this round of iterations. Vehicle 2 successfully contains the task f and finally gets the schedule b_2, f_2, g_2, d_2, because it predicts that f_1 is greater than f_2.

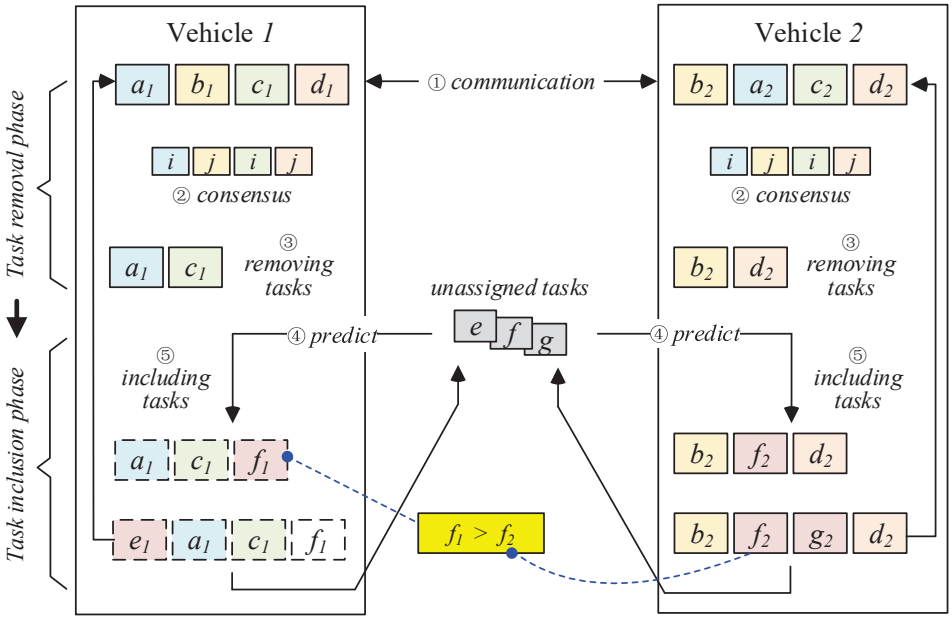


Figure 2. Diagram of distributed task scheduling method based on conflict prediction.

The specific algorithm design process is given below.

3.2. Task Removal

This subsection focuses on the design of the task removal strategy as shown in Algorithm 1. As described by the aforementioned basic idea, the significance γ_i of the tasks in the task list of a vehicle should be transmitted to other vehicles $v_j, j = 1, \ldots, n, j \neq i$ in order to notify them of removing the corresponding tasks from their scheduler, updating local significance list $\gamma_i^\diamond$ and to get the significance further decreased. Similar as in PI, a vehicle list $\boldsymbol{\beta}_i = [\beta_{i,1}, \ldots, \beta_{i,m}]^{\mathrm{T}}$ and a local significance list $\boldsymbol{\beta}_i^\diamond = [\beta_{i,1}^\diamond, \ldots, \beta_{i,m}^\diamond]^{\mathrm{T}}$ are also utilized in assistance with updating a proper global significance list, in which $\boldsymbol{\beta}_i$ notes the global assigned vehicle $\beta_{i,k}$ to each task t_k $(k = 1, \ldots, m)$ known by vehicle v_i, and $\boldsymbol{\beta}_i^\diamond$ represent similar descriptions that are local to vehicle v_i. If a vehicle receives a significance smaller than the one it has produced for a particular task already in its task list, this task is then required to be removed from the task list (Line 1). Next, find a set of tasks to be deleted from the task list via $\mathbf{d}_i \leftarrow \mathbf{p}_i[\boldsymbol{\beta}_i^\diamond[\mathbf{p}_i] \neq i]$, where the formula in square brackets is used to find the task index of the failed bid for vehicle v_i recorded in the local vehicle list $\boldsymbol{\beta}_i^\diamond$ (Line 2). Finally, the time cost of the remaining tasks in $\mathbf{p}_i$ and the significance of the remaining tasks in $\mathbf{d}_i$ are then updated due to the removal of tasks (Line 3). The algorithm then moves to the task inclusion phase as described previously.

Algorithm 1 Task removal procedure running on vehicle ν_i

Input: the global significance list γ_i, the global vehicle list $\boldsymbol{\beta}_i$, scheduler $\mathbf{p}_i$, local significance list $\gamma_i^\diamond$, local vehicle list $\boldsymbol{\beta}_i^\diamond$, the vehicle index i

Output: the updated $\mathbf{p}_i, \gamma_i^\diamond, \boldsymbol{\beta}_i^\diamond$

1: Perform the consensus procedure to update local significance list $\gamma_i^\diamond$ and local vehicle list $\boldsymbol{\beta}_i^\diamond$.
2: Remove tasks belonging to $\mathbf{d}_i \leftarrow \mathbf{p}_i[\boldsymbol{\beta}_i^\diamond[\mathbf{p}_i] \neq i]$.
3: Update time costs $c_{i,\kappa}(\mathbf{p}_i)$ and significance $\gamma_i^\diamond$ for the rest of tasks in $\mathbf{p}_i$ respectively.

3.3. Task Inclusion

This subsection presents an iterative task inclusion strategy based on conflict prediction. The UAV take the significance of the winner of each task in the previous round as the benchmark when they remove or include tasks. In other words, this part of the benchmark information could be already outdated. To deal with this problem, a prediction operation is added into the inclusion phase of PI to predict the inclusion loop operations that other vehicles are performing simultaneously. The details of the process are shown in Algorithm 2.

Algorithm 2 Task inclusion procedure running on vehicle ν_i

Input: the local scheduler $\mathbf{p}_i$, the global significance list γ_i, the global vehicle list $\boldsymbol{\beta}_i^\diamond$, the vehicle index i

Output: the updated $\mathbf{p}_i$, γ_i and $\boldsymbol{\beta}_i$

1: Initialize the set Ξ to be null, $\widetilde{\mathbf{p}}_j \leftarrow \mathbf{p}_j, \widetilde{\gamma}_j^\diamond \leftarrow \gamma_j$, for ν_j where $j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$.
2: **while** $|\mathbf{p}_i| \leq N_i$ **do**
3: Compute the marginal significance list $\gamma_{i,q}^\oplus$ according to Equation 3, for the tasks t_q, $q \in \{1, \ldots, m\}/\Xi$.
4: **if** $\max_{q \in \{1,\ldots,m\}/\Xi}\{\gamma_{i,q}^\diamond - \gamma_{i,q}^\oplus\} > 0$ **then**
5: $t_{q^*} \leftarrow \arg\max_{q \in \{1,\ldots,m\}/\Xi}\{\gamma_{i,q}^\diamond - \gamma_{i,q}^\oplus\}$ with corresponding position $l^* \leftarrow \arg\omega_q^\oplus(\mathbf{p}_i, t_{q^*})$.
6: $\widehat{\gamma}_{i,q^*} \leftarrow Predictor(t_{q^*}, i, \mathcal{N}; \widetilde{\mathbf{p}}_j, \widetilde{\gamma}_j^\diamond)$.
7: **if** $\gamma_{i,q^*}^\diamond < \widehat{\gamma}_{i,q^*}$ **then**
8: Insert task t_{q^*} into $\mathbf{p}_i$ at position l^*.
9: Update vehicle list's entry $\boldsymbol{\beta}_{i,l^*}^\diamond \leftarrow i$.
10: Update time costs $c_{i,\kappa}(\mathbf{p}_i)$ for the tasks followed after t_{q^*} in $\mathbf{p}_i$.
11: **end if**
12: **else**
13: break.
14: **end if**
15: $\Xi \leftarrow \Xi \cup \{t_{q^*}\}$.
16: **end while**
17: Update significance $\gamma_{i,q}$ and vehicle list $\boldsymbol{\beta}_{i,q}$, for all $q \in 1, \ldots, m$.
18: **return** $\mathbf{p}_i, \gamma_i, \boldsymbol{\beta}_i$.

First, define an empty set Ξ for storing tasks that cannot be inserted or tasks that have been inserted, and copy the scheduler $\mathbf{p}_j$ and significance list γ_j of adjacent vehicles to $\widetilde{\mathbf{p}}_j$ and $\widetilde{\gamma}_j^\diamond$, respectively (Line 1), where $\nu_j, j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$ indicates that the vehicle ν_j is interconnected with ν_i. Second, determine whether the current scheduler still has capacity to insert new tasks (Line 2), and if so, computing the marginal significance list $\gamma_{i,q}^\oplus$ for the tasks t_q, $q \in \{1, \ldots, m\}$ but $q \notin \Xi$ according to Equation (3) (Line 3). Next, the tasks are then continuously added to the scheduler $\mathbf{p}_i$ one after another (Lines 4–14) by each time satisfying the criteria,

$$\max_{q \in \{1,\ldots,m\}/\Xi}\{\gamma_{i,q}^\diamond - \gamma_{i,q}^\oplus\} > 0 \tag{5}$$

and leading to the largest reduction of the overall time cost with the new task

$$t_{q^*} \leftarrow \arg \max_{q \in \{1,\ldots,m\}/\Xi} \{\gamma^{\diamond}_{i,q} - \gamma^{\oplus}_{i,q}\} \tag{6}$$

at position $l^* \leftarrow \arg \omega^{\oplus}_{q}(\mathbf{p}_i, t_{q^*})$ in the task list.

Unlike PI, before vehicle ν_i inserts the task t_{q^*} at position l^*, we add a step (Line 6): an operator '*predictor*' is used to simulate adjacent vehicles $\nu_j, j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$ executes inclusion operation, to determine if the task t_{q^*} will also exist in the scheduler of the adjacent vehicle ν_j, and if so, compare the marginal significance $\gamma^{\oplus}_{i,q^*}$ of task t_{q^*} with the least predicted significance $\widehat{\gamma}_{i,q^*}$ of the adjacent vehicles. If $\gamma^{\diamond}_{i,q^*} < \widehat{\gamma}_{i,q^*}$, indicating that the significance of t_{q^*} on ν_i is smaller, then the task t_{q^*} will be confirmed to insert into vehicle ν_i's scheduler $\mathbf{p}_i$, otherwise the insertion will be abandoned (Lines 7–11). Details about the predictor will be described in Section 3.4.

Once the task t_{q^*} is indeed inserted, the l^*th element of the local vehicle list $\boldsymbol{\beta}^{\diamond}_{i,l^*}$ is set as i (Line 9), and $c_{i,\kappa}(\mathbf{p}_i)$ for the tasks followed after t_{q^*} in $\mathbf{p}_i$ need to be updated (Line 10). Before inserting the next task, add the task t_{q^*} that may have been inserted or abandoned to the set Ξ to avoid being reconsidered next time (Line 15). Finally, the significance list $\boldsymbol{\gamma}_i$ on the vehicle ν_i needs to be updated according to Equation (3) before this phase is completed.

The whole algorithm stops when no changes can be made both in the task inclusion phase and in the task removal phase for a while.

3.4. Predictor

The predictor will simulate a round of task removal and task inclusion operations for all adjacent vehicles, and determine whether the task intended to be included exists in the updated path of the adjacent vehicles. If so, the return value is the marginal significance of the input task on a certain vehicle, which is the smallest among all adjacent vehicles. The specific process as shown in Algorithm 3 is:

Algorithm 3 Predictor procedure running on vehicle ν_i

Input: the task intended to be removed t_{q^*}, the vehicle index i, the set $\mathcal{N}$; paths $\widetilde{\mathbf{p}}_j$, the local significance $\boldsymbol{\gamma}^{\diamond}_j$, for ν_j, where $j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$.

Output: the predicted significance $\widehat{\gamma}_{i,q^*}$ of task t_{q^*}.

1: Initialize $\widehat{\gamma}_{i,q^*}$ to a large initial value.
2: **for** each vehicle ν_j, where $j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$ **do**
3: $[\widetilde{\mathbf{p}}_j, \widetilde{\boldsymbol{\gamma}}^{\diamond}_j, \widetilde{\boldsymbol{\beta}}^{\diamond}_j] \leftarrow \mathit{Removal}(\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j, \widetilde{\mathbf{p}}_j, \widetilde{\boldsymbol{\gamma}}^{\diamond}_j, \widetilde{\boldsymbol{\beta}}^{\diamond}_j, j)$.
4: $[\widetilde{\mathbf{p}}_j, \widetilde{\boldsymbol{\gamma}}^{\diamond}_j, \widetilde{\boldsymbol{\beta}}^{\diamond}_j] \leftarrow \mathit{Inclusion}(\widetilde{\mathbf{p}}_j, \widetilde{\boldsymbol{\gamma}}^{\diamond}_j, \widetilde{\boldsymbol{\beta}}^{\diamond}_j, j)$ that do not contain the prediction operation (corresponding to lines 6 and 7 of Algorithm 2).
5: **if** $t_{q^*} \in \widetilde{\mathbf{p}}_j$ and $\widetilde{\gamma}^{\diamond}_{j,q^*} < \widehat{\gamma}_{i,q^*}$ **then**
6: $\widehat{\gamma}_{i,q^*} \leftarrow \widetilde{\gamma}^{\diamond}_{j,q^*}$
7: **end if**
8: **end for**
9: **return** $\widehat{\gamma}_{i,q^*}$.

First, define a predicted significance $\widehat{\gamma}_{i,q^*}$ for task t_{q^*} on vehicle ν_i and initialize it to a large initial value (Line 1). Then, traverse all adjacent vehicles ν_j, where $j \in \mathcal{N}; j \neq i; g_{j,i}(t) = 1$ (Line 2), and sequentially call Algorithm 1 (Line 3) and Algorithm 2 (Line 4) to simulate each vehicle ν_j to complete a round of the task removal and task inclusion operations, thereby obtaining the predicted scheduler $\widetilde{\mathbf{p}}_j$ and the marginal significance $\widetilde{\boldsymbol{\gamma}}^{\diamond}_j$. It is worth noting that, to avoid infinite nesting of predictions, the include operation here will not include the prediction operation as shown in Algorithm 2, lines 6 and 7. Finally, it is judged whether the task t_{q^*} intended to be included exists in the predicted scheduler $\widetilde{\mathbf{p}}_j$.

If so, the predicted marginal significance $\widetilde{\gamma}^{\diamond}_{j,q^*}$ is compared with $\widehat{\gamma}_{i,q^*}$, and the smaller one is assigned to $\widehat{\gamma}_{i,q^*}$.

4. Communication Models

4.1. Distributed Implementation

We use a mobile ad hoc network (MANET) as a wireless communication facility between multiple drones. Ad-hoc networks are a special case of wireless networks in which each node acts as a router or access point by itself, without the help of infrastructure. With its characteristics of supporting mobile nodes, no network infrastructure, convenient network construction, self-organization, and good robustness, it is very suitable for the technical requirements of UAV swarms networking.

The specific implementation approach of the proposed method under networking conditions is as follows (as shown in Figure 3): (1) Construct a networking simulation platform mainly composed of the protocol stack and channel simulation, in which the protocol stack mainly adopts the transport layer protocol, network layer protocol and data link layer protocol running on the actual physical system, and channel simulation mainly simulates the loss of imperfect communication channels such as packets, delay, and noise floor. (2) Realize background operations for creating or destroying virtual network nodes, importing and forwarding external service data to virtual nodes, providing service data distribution interruption management, and calculating path loss based on node locations. (3) Implement the graphical user interface (GUI) to provide users with interface access functions such as system control, node control, and link configuration of the virtual network. (4) Each scheduler equipped with a replica of the proposed method is placed on the corresponding vehicle and communicates with other schedulers by binding virtual network nodes to negotiate the overall scheduling of the entire system.

Remark. Since the proposed method adopts a synchronous communication protocol, an additional synchronization signal is added when the service data is packaged for mutual waiting between scheduling rounds.

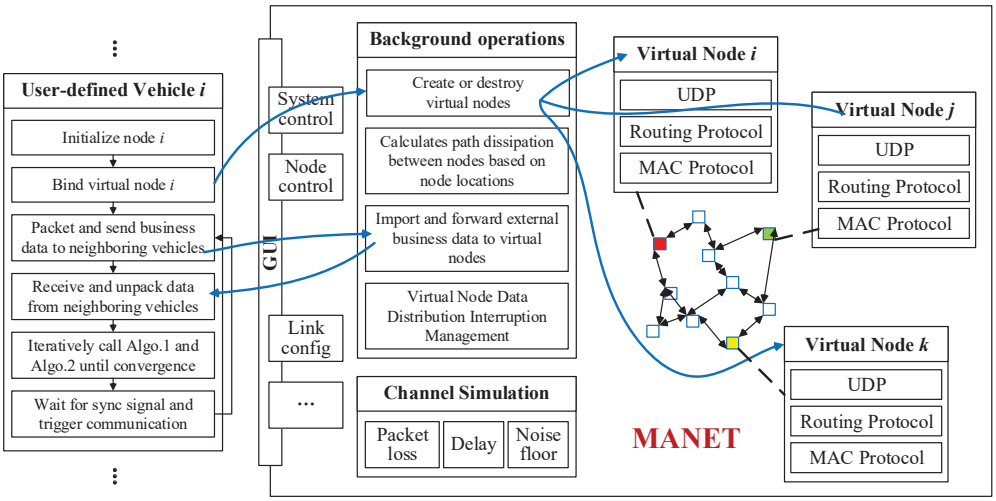


Figure 3. Schematic diagram of the distributed synchronization scheduling algorithm test platform based on MANET.

4.2. Protocol Stack

The core of MANET is the protocol stack, which mainly includes the MAC protocol of the data link layer, the routing protocol of the network layer, and the TCP/UDP protocol of the transport layer.

The MAC protocol is responsible for the management of network access and node resource utilization and is a key factor for the network to achieve high performance.

Commonly used MAC protocols are Carrier Sense Multiple Access with Collision Avoid (CSMA/CA), Time Division Multiple Access (TDMA), and so on. Literature [29] shows that the end-to-end delay and network throughput of CSMA/CA protocol increase with the increase of the number of network nodes; while the end-to-end delay and network throughput of TDMA protocol remain stable. That is to say, compared with CSMA/CA, TDMA has shorter end-to-end latency and higher throughput when the number of nodes is large. Then, for scalable UAV swarms, we prefer to choose TDMA as the MAC protocol for mobile ad hoc networks.

Routing protocols are roughly divided into the prior protocols (or table-driven protocols) and the reactive protocols (or on-demand routing protocols). The former requires each node to maintain a routing table, which is updated instantly when the network topology changes. Since there is no need to search for a route, the transmission delay is small, but the maintenance cost of the route is relatively large. Common protocols include Destination Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), Better Approach To Mobile Ad-Hoc Networking (batman), etc. The latter is a protocol for finding routes before transmission. Since there is no need to maintain the routing table, the overhead of maintaining the protocol is small, but the transmission delay is relatively large. Common protocols include Dynamic Source Routing (DSR), Ad hoc On-Demand Distance Vector Routing (AODV), Temporally Ordered Routing Algorithm (TORA), etc. This work selects the batman-adv protocol, which runs on the data link layer in the form of a Linux kernel module, to realize data transparent transmission through MAC addressing instead of traditional IP addressing. This routing protocol is suitable for drone swarms, which usually have requirements on the power consumption and size of the ad-hoc network electronics.

For the transport layer protocol, we tend to choose UDP (User Datagram Protocol) in broadcast mode instead of TCP (Transmission Control Protocol). Although TCP provides reliable data transmission, we use UDP as it allows for broadcasting which is inherently efficient for consensus algorithms. Furthermore, ref. [30] shows that consensus times for CBBA under TCP and UDP unicast mode are far longer compared to UDP broadcast.

4.3. Communication Channel Simulation

We characterize imperfect physical communication channels by packet loss, latency, and noise floor. Reference [24] compares Bernoulli [31], Gilbert-Elliot (G.E.) [32,33] and Rayleigh Fading [34] models to simulate packet loss between nodes. These models account for path loss, fading, burst errors, and bandwidth saturation inherent in many communication channels. The Bernoulli model uses the parameter p ($0 \leq p \leq 1$) to set the probability that the destination node successfully receives the message from the source node; the G.E. model can be used to simulate the packet loss caused by bit errors and bandwidth saturation; the Rayleigh fading model predicts the attenuation of the received signal by assuming that the signal's amplitude will vary according to a Rayleigh distribution. We adopt the Rayleigh fading model to simulate imperfect communication environments as it comes closer to modeling realistic environments compared to Bernoulli or Gilbert-Elliot. The Rayleigh Fading model decides whether to discard a message by judging whether the total received power P_R of the destination node is less than the user-specified sensitivity threshold P_S. The total received power is obtained with the formula $P_R = P_T - P_L$, where P_T is the transmitted power and P_L is the total attenuation, which is given by $P_L = P_F + P_{PL}$. P_F stands for random path fading, which is a process in which the wireless signal attenuates and changes due to the interference of objects in the environment (such as buildings, trees, etc.). These objects cause the wireless signal to travel along multiple paths, each of which experiences different offsets in amplitude, frequency, and ultimately creates constructive or destructive interference at the receiver. To quantify P_F, an inverse discrete Fourier transform (IDFT) is usually used to obtain a Rayleigh random variable sequence, which is then sampled, and the resulting power value is converted to a decibel value as P_F. P_{PL} is the path loss due to distance with the formula $P_{PL} = P_{L_0} + 10\gamma \log_{10}(d/d_0)$ where d is the current distance between nodes, γ is the path loss exponent, and P_{L_0} is the path loss at the

reference distance d_0. Latency is achieved by stuffing packets into the delayed work path when the underlying data is exchanged.

5. Results and Discussion

This section analyzes and compares the Monte Carlo simulation results of the proposed method compared to the PI algorithm based on the MANET simulation platform.

5.1. Scenario and Environmental Setup

The scenarios adopted in this paper are consistent with that of literature [3,8]. The setup uses a rescue team equally split that two vehicles need to rescue two types of survivors, who are likewise equally split into two types that require food and medicines, respectively. Parameters related to the scenario are summarized in Table 1 and are explained as follows. The vehicles are randomly distributed in a 10 km × 10 km × 0 km ground space, where the vehicles' speeds are assumed to be constant and are set to 30 m/s and 50 m/s, respectively. The tasks take place in a 3-D space spanning 10 km × 10 km × 1 km, with coordinates drawn from uniform distributions randomly. The medicine tasks last for a duration of 300 s and the food tasks last for 350 s. The deadlines for starting each rescue are uniformly distributed on a timeline between the 0 and 2000 s. Given the random initialization of tasks and vehicle locations and deadlines, sometimes it is impossible for some tasks to be started by any vehicle before their deadlines. In these simulations, all the task information is available to all the vehicles up front.

Table 1. Parameters related to the simulation scenario.

Properties	Medicine	Food
Vehicle speed	30 m/s	50 m/s
Vehicle initial position	[10 km × 10 km × 0 km]	[10 km × 10 km × 0 km]
Task duration	300 s	500 s
Task deadline	[0, 2000]s	[0, 2000]s
Task location	[10 km × 10 km × 1 km]	[10 km × 10 km × 1 km]

All algorithms are evaluated in a simulation environment built based on the ad-hoc network model of Section 4, as shown in Figure 4. Among them, UDP, dynamic TDMA, and Batman-adv are used to form a protocol stack, the Bernoulli model is used to simulate packet loss, and the Rayleigh Fading model is used to simulate distance-based path attenuation. Delayed work paths for TDMA to simulate delays. Unlike previous works that simulate varying communication with different network topologies (such as a row, star, circular, mesh, etc.), this paper assumes a full mesh topology, in which every vehicle attempts to communicate with every other vehicle by continuously broadcasting messages. This assumption facilitates the discovery of performance differences due to applying the proposed method as opposed to changing the vehicles' network topology. Parameters related to the communication are summarized in Table 2.

Table 2. Parameters related to communication simulation.

Properties	Values
Network topology range	15 Km
Bandwidth transmission	2 Mbps
Networking mode	dynamic TDMA
Transmitted power P_T	30 dB
Sensitivity threshold P_S	−25 dB
Path loss P_{L_0} at $d_0 = 1$ [1]	40 dB
Path loss exponent γ [2]	2.5

[1] Calculated using the Friis propagation model [35] assuming a 2.4 GHz signal commonly used in many communication networks. [2] Simulate semi-urban to rural environments (γ ranges between 2 to 6 with 2 for uncluttered space and 6 for densely obstructed urban areas [36]).

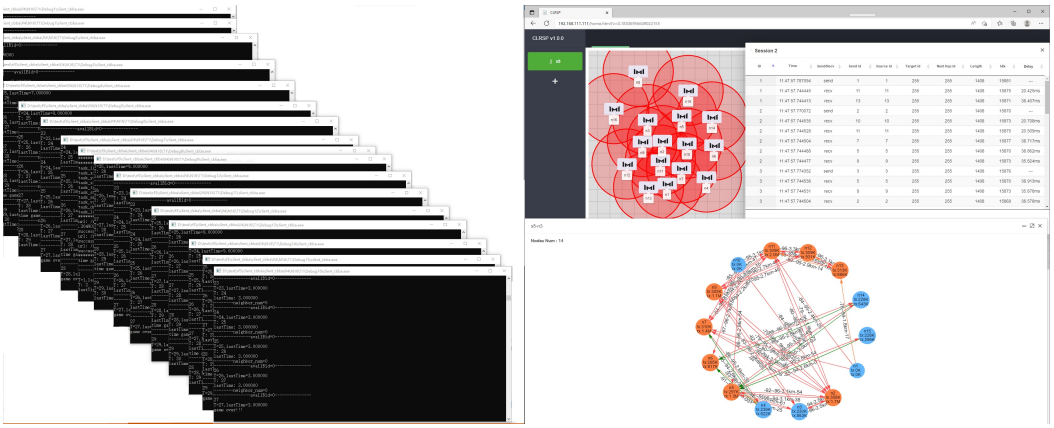


Figure 4. Simulation platform based on the ad-hoc network model.

5.2. Results and Analysis

This section presents the simulation results of using the proposed method denoted as PI-Predict and PI to solve the above search-and-rescue problem, and analyzes the advantages of the algorithm from the perspectives of global optimization and convergence. We use the Monte Carlo method to implement random simulations, where each method under the same condition of every $p \in \{2, \ldots, 6\}$ and $n \in \{6, 8, \ldots, 20\}$ is tested 1000 times. p is the ratio of the number of tasks and drones, and n is the number of drones.

The optimization of the algorithm is discussed below, including the average time cost and the total number of assignments. Figure 5 shows the reduction ratio of the average time cost of PI-Predict relative to PI under the same allocation number with bars representing 40 sets of samples under different combinations of p and n. The positive bar indicates that the decrease ratio of the average time cost is higher than the increase ratio. The opposite is true for negative bar. The reason why the average time cost increases rather than decreases is that it is affected by the distribution of vehicles-tasks. Affected by the distribution of vehicles-tasks, the scheduling results generated by the proposed strategy have certain randomness. From the 40 data sets, 28 data sets show that the proposed method can achieve lower average time cost with more distribution of vehicles-tasks. From the overall trend, when $p$is small, the number of tasks that can be assigned is limited, the performance of the algorithm varies greatly with different random distribution and algorithm strategy change, The optimization of scheduling results is also different. For example, when $p = 2$, there is little difference in the number of positive and negative bars, which is caused by randomness; When the number of tasks that can be assigned increases with the increase of p, PI-Predict shows a more stable trend of improving the optimization performance, and the proportion of improvement is larger. For example, when $p = 6$, the color proportion in the positive bar shows an increasing trend, indicating that PI-Predict provides a stable performance improvement.

In addition to the average time cost, PI-Predict increases the number of assigned tasks under most of the random distributions of vehicles-tasks. From the 40 groups of Monte Carlo simulation data generated by different combinations of p and n in Figure 6, it can be seen that PI-predict can bring different degrees of increase in the allocation number. Due to the randomness of scheduling results caused by the random distribution of vehicles-tasks, it is natural that the allocation number may decrease rather than increase. The color bar in the figure represents the proportion of simulation times with increasing allocation number in 1000 simulations under a certain combination of p and n, and the gray part of the bar represents the proportion of simulation times with decreasing allocation number (for example, when $p = 2$ and $n = 6$, the growth ratio is about 20% and the reduction ratio is less than 10%, implying that the remaining 70% of allocation number remains unchanged). In terms of the overall trend, with the increase of p, the number of tasks that can be assigned increases, and the number of tasks assigned also increases accordingly. When p is the same,

as n increases, the number of tasks that can be assigned will also increase, and the number of tasks assigned will also increase.

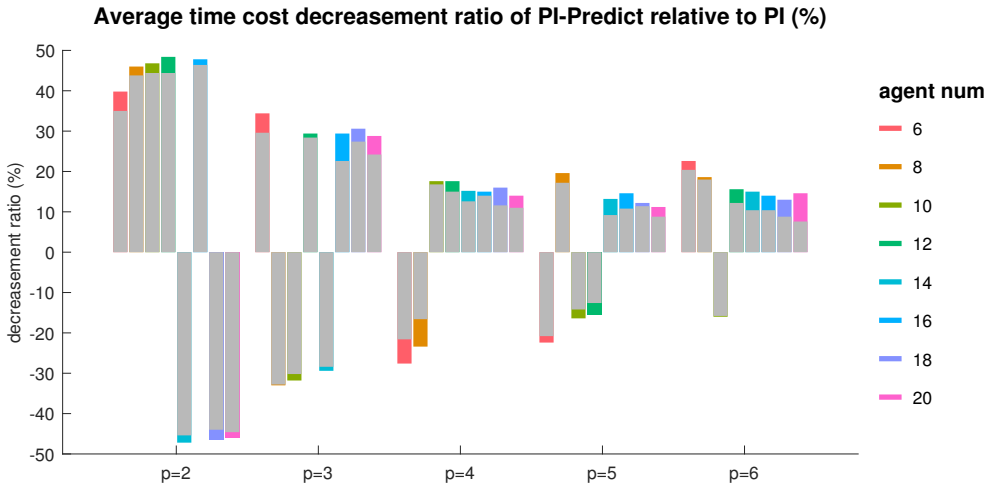


Figure 5. Average time cost reduction ratio of PI-Predict relative to PI.

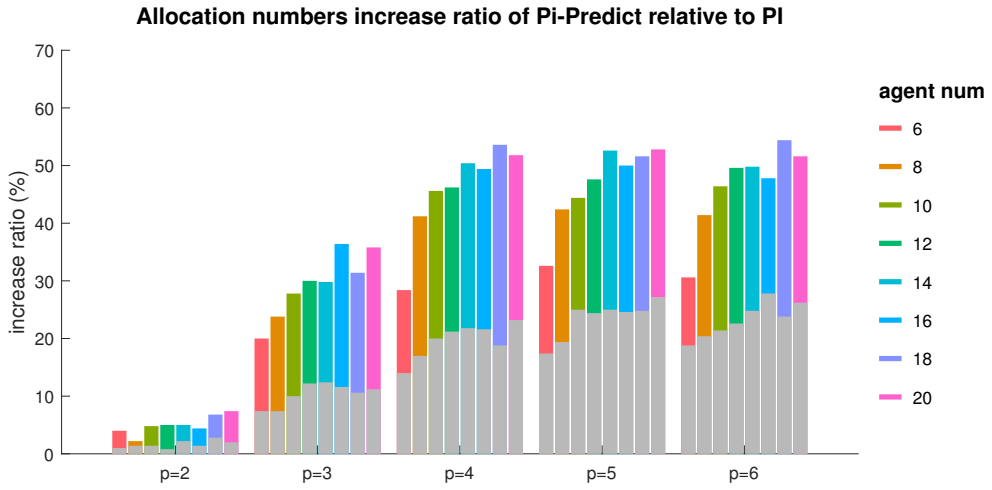


Figure 6. Allocation numbers increase ratio of PI-infer-conv relative to PI.

Next, we discuss the efficiency of the algorithm, including the deadlock ratio and the number of iterations.

Figure 7 summarizes the deadlock ratios of PI and PI-Predict under different p, which are calculated by the ratio of the number of deadlocks to the number of Monte Carlo simulations. Among them, the number of deadlocks is the sum of all deadlock cases in the range of $n \in \{2, 4, \ldots, 20\}$, and the number of Monte Carlo simulations is 1000. The potential reason for the reduction of the deadlock ratio is that the iterative inclusion strategy based on conflict prediction can eliminate the deadlock that occurs between two adjacent rounds, but it cannot do anything about the deadlock beyond the adjacent rounds, which can usually only be tackled with deliberate algorithm truncation.

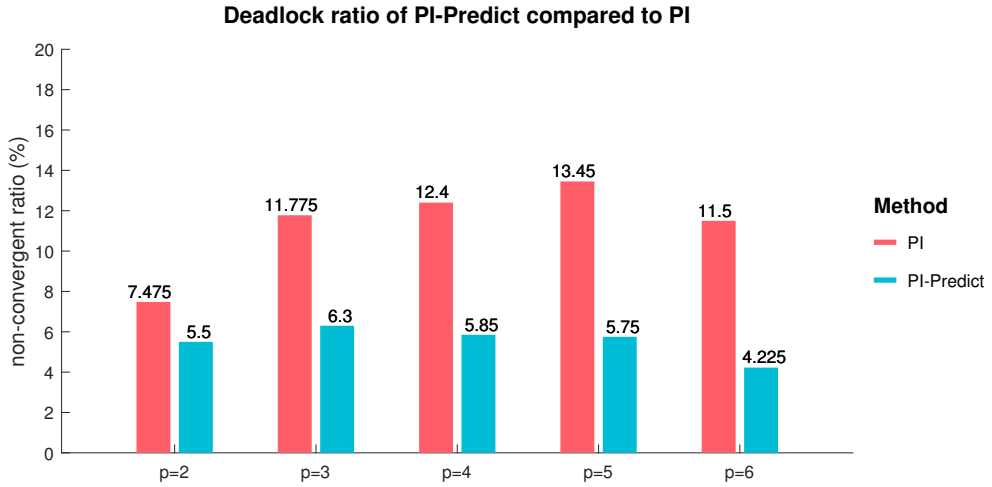


Figure 7. Deadlock ratio of PI-Predict compared to PI.

Figure 8 shows the box plots of the iterations number for PI and PI-Predict under the condition of $p = 6$. The statistical results of the algorithm are similar when p is equal to other values, and will not be repeated here. As can be seen from the figure, the number of iterations of PI-Predict has been greatly reduced compared to PI, which is caused by the conflict prediction mechanism. It is worth noting that both algorithms have outliers when the number of iterations is equal to 100, which is the number of algorithm truncations we set to deal with the deadlock problem.

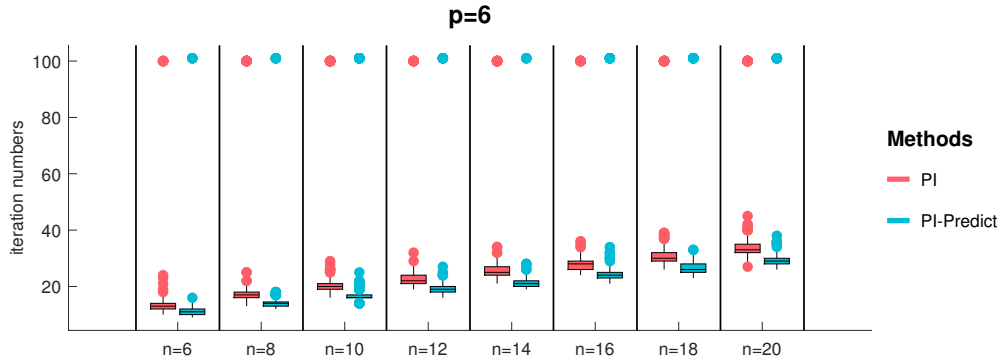


Figure 8. Box plots of iterations number for PI-Predict compared with PI.

From the above analysis and discussion, compared with PI, the proposed method can reduce the average time cost and increase the total allocation number under most of the random distributions of vehicles and tasks, and significantly reduce the deadlock ratio and the number of iteration rounds.

6. Conclusions

This paper modifies the baseline PI algorithm by integrating a new removal strategy and a conflict prediction operator into the tasks removal phase and inclusion phase of the algorithm respectively. The proposed method iterates between a communication and consensus phase, and a task inclusion phase, the first phase being used to include tasks into a vehicle's scheduler while the latter being responsible for the consensus on significance of tasks for each vehicle and for removing the tasks that have been taken over by other vehicles. In the communication and consensus phase, the removal strategy frees up more local space, making it possible for more new tasks to be added. This essentially increases the exploratory nature of the algorithm to improve the overall optimality of the multi-task scheduling problem. In the task inclusion phase, the conflict prediction can reduce the

number of iterations by predicting the task removal and task inclusion operation of neighbor vehicles synchronously, predicting and resolving potential allocation conflicts in advance. Furthermore, the operator can also avoid some deadlock that occur in adjacent rounds and reduce the deadlock ratio, which is of great benefit to UAV swarm task scheduling. Finally, Monte Carlo simulation experiments on the ad-hoc network simulation platform were carried out, and a large number of results show that, compared with PI, the proposed method can reduce the average time cost and increase the total allocation number under most of the random distributions of vehicles-tasks, and significantly reduce the deadlock ratio and the number of iteration rounds.

In the future, the main following research directions will be: first, focusing on exploration and utilization, to carry out special research on algorithm performance; Secondly, the consensus rules of combinatorial optimization under communication constraints would be studied [37]. Third, the algorithm verification of communication or flight control hardware in loop should be carried out at appropriate time.

Author Contributions: Conceptualization, J.L. and R.C.; methodology, J.L.; software, J.L.; validation, R.C.; investigation, J.L.; writing—original draft preparation, J.L.; writing—review and editing, R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Science and Technology Innovation 2030-Key Project of 'New Generation Artificial Intelligence' under Grant 2020AAA0108200.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Wang, X.; Yang, S.; Guo, Z.; Lian, M.; Huang, T. A Distributed Dynamical System for Optimal Resource Allocation over State-Dependent Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 2940–2951. [CrossRef]
2. Korsah, G.A.; Ayorkor, G.; Stentz, A.; Dias, M.B. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robot. Res.* **2013**, *32*, 1495–1512. [CrossRef]
3. Zhao, W.; Meng, Q.; Chung, P. A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario. *IEEE Trans. Cybern.* **2015**, *46*, 902–915. [CrossRef]
4. Geng, N.; Meng, Q.; Gong, D.; Chung, P.W. How Good are Distributed Allocation Algorithms for Solving Urban Search and Rescue Problems? A Comparative Study With Centralized Algorithms. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 478–485. [CrossRef]
5. Ganguly, S. Multi-objective distributed generation penetration planning with load model using particle swarm optimization. *Decis. Mak. Appl. Manag. Eng.* **2020**, *3*, 30–42. [CrossRef]
6. Stanković, A.; Petrović, G.; Ćojbašić, Ž.; Marković, D. An application of metaheuristic optimization algorithms for solving the flexible job-shop scheduling problem. *Oper. Res. Eng. Sci. Theory Appl.* **2020**, *3*, 13–28. [CrossRef]
7. Zhang, K.; Collins, E.G., Jr.; Shi, D. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Trans. Auton. Adapt. Syst.* **2012**, *7*, 21:1–21:22. [CrossRef]
8. Whitbrook, A.; Meng, Q.; Chung, P. 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.
9. Bandyopadhyay, S.; Chung, S.J.; Hadaegh, F.Y. Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents. *IEEE Trans. Robot.* **2017**, *33*, 1103–1123. [CrossRef]
10. Omidshafiei, S.; Agha–Mohammadi, A.A.; Amato, C.; Liu, S.Y.; How, J.P.; Vian, J. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *Int. J. Robot. Res.* **2017**, *36*, 231–258. [CrossRef]
11. Chen, C.-H.; Chou, F.-I.; Chou, J.-H. Optimization of robotic task sequencing problems by crowding evolutionary algorithms. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *52*, 6870–6885. [CrossRef]
12. Kim, S.; Moon, I. Traveling salesman problem with a drone station. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *49*, 42–52. [CrossRef]
13. Xin, B.; Wang, Y.; Chen, J. An efficient marginal-return-based constructive heuristic to solve the sensor-weapon-target assignment problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *49*, 2536–2547. [CrossRef]
14. Chen, W.; Wang, D.; Li, K. Multi-user multi-task computation offloading in green mobile edge cloud computing. *IEEE Trans. Serv. Comput.* **2018**, *12*, 726–738.

15. Mnif, S.; Elkosantini, S.; Darmoul, S.; Said, L.B. An immune network based distributed architecture to control public bus transportation systems. *Swarm Evol. Comput.* **2019**, *50*, 100478. [CrossRef]
16. Mudrova, L.; Hawes, N. Task scheduling for mobile robots using interval algebra. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 383–388.
17. Testa, A.; Rucco, A.; Notarstefano, G. Distributed mixed-integer linear programming via cut generation and constraint exchange. *IEEE Trans. Autom. Control.* **2019**, *65*, 1456–1467. [CrossRef]
18. Camisa, A.; Notarnicola, I.; Notarstefano, G. Distributed primal decomposition for large-scale milps. *IEEE Trans. Autom. Control.* **2021**, *67*, 413–420. [CrossRef]
19. Choi, H.-L.; Brunet, L.; How, J.P. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. Robot.* **2009**, *25*, 912–926. [CrossRef]
20. Buckman, N.; Choi, H.-L.; How, J.P. Partial replanning for decentralized dynamic task allocation. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; p. 0915.
21. Johnson, L.; Choi, H.L.; Ponda, S.; How, J.P. Allowing non-submodular score functions in distributed task allocation. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Maui, HI, USA, 10–13 December 2012.
22. Johnson, L.; Ponda, S.; Choi, H.; How, J. Improving the Efficiency of a Decentralized Tasking Algorithm for UAV Teams with Asynchronous Communications. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Toronto, ON, Canada, 2–5 August 2010.
23. Ismail, S.; Liang, S. Decentralized hungarian-based approach for fast and scalable task allocation. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017.
24. Nayak, S.; Yeotikar, S.; Carrillo, E.; Rudnick-Cohen, E.; Jaffar, M.K.M.; Patel, R.; Azarm, S.; Herrmann, J.W.; Xu, H.; Otte, M. Experimental comparison of decentralized task allocation algorithms under imperfect communication. *IEEE Robot. Autom. Lett.* **2020**, *5*, 572–579. [CrossRef]
25. Carrillo, E.; Yeotikar, S.; Nayak, S.; Jaffar, M.K.M.; Azarm, S.; Herrmann, J.W.; Otte, M.; Xu, H. Communication-aware multi-agent metareasoning for decentralized task allocation. *IEEE Access* **2021**, *9*, 98712–98730. . [CrossRef]
26. Alighanbari, M.; How, J.P. Decentralized task assignment for unmanned aerial vehicles. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 12–15 December 2005; pp. 5668–5673.
27. Dionne, D.; Rabbath, C.A. Multi-UAV decentralized task allocation with intermittent communications: The DTC algorithm. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 5406–5411.
28. Kwa, H.L.; Kit, J.L.; Bouffanais, R. Balancing Collective Exploration and Exploitation in Multi-Agent and Multi-Robot Systems: A Review. *Front. Robot. AI* **2022**, *8*, 771520. [CrossRef]
29. Dong, Y.; Dong, Y.; Xiaojia, X.; Jie, L.; Zhenyi, L. Research on Ad-hoc Network for UAV Swarm Based on OPNET Simulation. In Proceedings of the 2020 IEEE 20th International Conference on Communication Technology (ICCT), Nanning, China, 28–31 October 2020; pp. 678–686. [CrossRef]
30. Rantanen, M.; Mastronarde, N.; Hudack, J.; Dantu, K. Decentralized task allocation in lossy networks: A simulation study. In Proceedings of the 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Boston, MA, USA, 10–13 June 2019; pp. 1–9.
31. Otte, M.; Kuhlman, M.J.; Sofge, D. Auctions for multi-robot task allocation in communication limited environments. *Auton. Robots* **2020**, *44*, 547–584. [CrossRef]
32. Gilbert, E.N. Capacity of a burst-noise channel. *Bell Syst. Tech. J.* **1960**, *39*, 1253–1265. [CrossRef]
33. Elliott, E.O. Estimates of error rates for codes on burst-noise channels. *Bell Syst. Tech. J.* **1963**, *42*, 1977–1997. [CrossRef]
34. Zheng, Y.R.; Xiao, C. Simulation models with correct statistical properties for Rayleigh fading channels. *IEEE Trans. Commun.* **2003**, *51*, 920–928. [CrossRef]
35. Khattak, R.; Chaltseva, A.; Riliskis, L.; Bodin, U.; Osipov, E. Comparison of wireless network simulators with multihop wireless network testbed in corridor environment. In *International Conference on Wired/Wireless Internet Communications*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 80–91.
36. Rappaport, T.S. *Wireless Communications: Principles and Practice*; Prentice Hall PTR: Hoboken, NJ, USA, 1996; Volume 2.
37. Komareji, M.; Shang, Y.; Bouffanais, R. Consensus in topologically interacting swarms under communication constraints and time-delays. *Nonlinear Dyn.* **2018**, *93*, 1287–1300. [CrossRef]