

## Article

# A Blockchain-Based Multi-Unmanned Aerial Vehicle Task Processing System for Situation Awareness and Real-Time Decision

Ziqiang Chen <sup>1</sup>, Xuanrui Xiong <sup>1,\*</sup>, Wei Wang <sup>2</sup>, Yulong Xiao <sup>1</sup> and Osama Alfarraj <sup>3</sup>

<sup>1</sup> School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

<sup>2</sup> School of Software, Dalian University of Technology, Dalian 116024, China

<sup>3</sup> Computer Science Department, Community College, King Saud University, Riyadh 11437, Saudi Arabia

\* Correspondence: xiongrx@cqupt.edu.cn; Tel.: +86-139-8327-2758

**Abstract:** With the rapid advancement of Unmanned Aerial Vehicle (UAV) technology, UAV swarms are being extensively applied in various fields, such as intelligent transportation, search and rescue, logistics delivery, and aerial mapping. However, the utilization of UAV swarms in sustainable transportation also presents some challenges, such as inefficient task allocation and data transmission security issues, highlighting the importance of privacy protection in this context. To address these issues, this study applies blockchain technology to multi-UAV tasks and proposes a blockchain-based multi-UAV task processing system for situation awareness and real-time decisions. The primary objective of this system is to enhance the efficiency of UAV swarm task scheduling, bolster data transmission security, and address privacy protection concerns. Utilizing the highly secure features of blockchain technology, the system constructs a distributed task processing network. System tasks are stored in the blockchain through smart contracts, ensuring the immutability and verifiability of task information. Smart contracts have an automatic execution capability, whereby the system can efficiently coordinate tasks and maintain the consistency of task execution information through consensus mechanisms. Additionally, adopting the Pointer Network structure for intelligent path planning based on task allocation results leads to the attainment of the shortest service routes, consequently expanding the service coverage of sustainable transportation systems while reducing energy consumption. This further advances the realization of urban sustainable transportation. Through experimental results, we verify that the proposed system enables real-time task scheduling and collaborative processing for multiple UAVs, significantly enhancing the efficiency, security, and privacy protection level of UAV swarm task execution in the context of sustainable transportation. It makes a positive contribution to building more sustainable urban transportation systems.

**Keywords:** UAV; blockchain; task scheduling; smart contracts; privacy protection; path planning



**Citation:** Chen, Z.; Xiong, X.; Wang, W.; Xiao, Y.; Alfarraj, O.

A Blockchain-Based Multi-Unmanned Aerial Vehicle Task Processing System for Situation Awareness and Real-Time Decision. *Sustainability* **2023**, *15*, 13790. <https://doi.org/10.3390/su151813790>

Academic Editor: Giovanni Leonardi

Received: 24 July 2023

Revised: 12 September 2023

Accepted: 13 September 2023

Published: 15 September 2023



**Copyright:** © 2023 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, the rapid development of Internet of Things (IoT) technology has driven the trend of the digital era [1]. As an integral component of the IoT, UAVs have progressively found applications in various domains of human life, including intelligent transportation, agriculture, logistics, and more [2–4]. This has garnered significant attention from both academia and industry [5]. Simultaneously, with the advancement of communication technology, an increasing number of mobile devices have integrated into edge networks, resulting in a substantial surge in the generation of real-time service requests [6]. This, in turn, poses security threats to a considerable amount of data distributed across the network [7]. In the context of multi-UAV missions, the encompassed data extends beyond mere positional and environmental information to encompass task directives, image data, and other pertinent sources. The real-time precision and accuracy of these data play

a pivotal role in the system's overall performance. Nevertheless, the imperative collaboration among geographically dispersed UAVs necessitates data sharing to facilitate precise environmental perception and swift decision-making. This engenders concerns over data privacy and security, especially within a distributed setting where data may become vulnerable to unauthorized access and tampering. Furthermore, conventional centralized data management methodologies may fall short in addressing the exigent real-time demands of multi-UAV missions, given their susceptibility to network congestion and single-point failures, among other challenges.

Building upon the aforementioned backdrop, we propose the design of a blockchain-based multi-UAV mission processing system for situational awareness and real-time decision-making. The objective of this system is to leverage blockchain technology to establish a secure and real-time platform for processing multi-UAV missions. By doing so, the system aims to enhance the efficacy of task allocation and flight path planning for UAVs, facilitate collaborative mission execution, mitigate concerns over data privacy and security, facilitate real-time data sharing, and offer efficient decision-making support. This innovative approach is poised to unlock substantial potential and promising applications within the realm of UAV technology, propelling it toward greater development and prospects.

### *1.1. Research Challenges*

The inherent constraints posed by the dynamic wireless channel, limited battery capacity, and restricted computational resources of UAVs have highlighted the inefficiencies of conventional methods in the realm of UAV-connected Internet [8]. In contrast to the confined sensing coverage and communication capabilities exhibited by solitary UAVs, the orchestrated efforts of multiple UAVs have paved the way for enhanced sensing and transmission services [9]. With the burgeoning expansion of UAV systems, the realm of collaborative processing for numerous UAV tasks encounters a spectrum of challenges. The wireless and ever-shifting landscape of communication deployment exposes UAVs to an array of security vulnerabilities, including network attacks and physical breaches [10]. The copious data amassed during UAV operations brings forth privacy apprehensions, and conventional centralized systems might be susceptible to perils such as data manipulation, unauthorized data access, or data leakage. In response to these formidable challenges, the fusion of blockchain technology emerges as a solution, ensuring data encryption, fortifying sensitive data security, averting data leakage, and upholding the security and privacy tenets of UAV systems.

The execution of UAV missions frequently necessitates stringent real-time and low-latency prerequisites. Nonetheless, the consensus algorithm and the subsequent transaction verification procedure within a blockchain network could potentially introduce heightened latency. Consequently, the pursuit of expeditious task allocation and prompt decision-making within a blockchain framework, aimed at meeting the exigent low-latency prerequisites of UAV mission processing, poses a formidable challenge. Moreover, the deployment and operational expenditures linked with blockchain technology might be substantial, particularly within expansive multi-UAV mission processing frameworks. Addressing the research quandary of curtailing the expenses associated with blockchain systems, while concurrently harmonizing the interplay between costs and advantages, becomes imperative to attain the economic viability of multi-UAV mission processing systems.

### *1.2. Contributions*

To address the aforementioned issues, we propose a blockchain-based multi-UAV task processing system for situation awareness and real-time decisions. The system is designed to ensure data security and privacy protection, enhance task responsiveness, and enable intelligent decision support. This will bring more opportunities and potential to the field of UAV applications while improving the efficiency and security of mission execution. The main contributions of this work are as follows:

- By introducing blockchain technology, we achieve secure transmission and storage of data. The decentralized and encrypted nature of blockchain protects the integrity and confidentiality of task data, preventing unauthorized access and tampering, thereby enhancing the system's data security and privacy protection capabilities;
- Our system has the capability to acquire and analyze network data in real-time, including sensor data and communication data. Based on these data, the system utilizes intelligent algorithms to determine task priorities and allocate optimal resources, enabling efficient task processing and optimized resource utilization. This enhances task allocation efficiency and improves overall task execution effectiveness;
- We have established a decentralized collaboration platform that enables direct communication and information sharing among UAVs. Through the use of smart contracts and distributed consensus mechanisms, the system is capable of facilitating trustworthy task collaboration and resource allocation, thereby enhancing the collaborative processing capabilities of multi-UAV systems.

This paper is organized as follows: In Section 2, a review of related work is presented. Section 3 introduces the system design. The implementation of the system and its performance analysis are described in Section 4. Finally, Section 5 concludes the paper. For abbreviations of terminology in the article, refer to Table 1.

**Table 1.** Nomenclature table.

Term	Description
UAV	An aircraft operated without a human pilot on board, commonly known as a drone, used for various tasks.
UAV Swarm	A group of UAVs that work together to achieve common goals through coordinated and allocated tasks.
Blockchain Technology	A decentralized and secure digital ledger technology used for data integrity, security, and sharing.
IoT	A network of interconnected devices that can exchange data and communicate over the internet.
Frontend	The user interface and presentation layer of the system.
Backend	The server-side logic and data processing layer of the system.
Vue.js	A JavaScript framework for building user interfaces.
Spring Cloud	A framework for building distributed systems and microservices.
Spring Boot	A framework for building stand-alone, production-grade applications.
MyBatis	MyBatis is a Java-based persistence framework that simplifies database interactions. It provides an SQL mapping approach, allowing developers to define queries in XML or annotations, making it easier to manage and retrieve data from relational databases.
FastAPI	A modern Python-based web framework known for high performance.
MAVSDK	MAVLink Drone SDK, used for drone hardware scheduling.
MySQL	MySQL is an open-source RDBMS renowned for its speed, reliability, and SQL-based interaction. It is used for data storage, management, and retrieval, catering to various projects from personal to enterprise.
Redis	Redis is an open-source, high-performance, in-memory data store used for caching and messaging. It supports diverse data structures and is commonly employed for caching and real-time data management.
TSP	TSP is a classic optimization problem where the goal is to find the shortest possible route that visits a set of given cities and returns to the starting city. It is a common problem in the field of combinatorial optimization.
NP-hard	NP-hard refers to problems that are at least as hard as the hardest problems in the NP complexity class. These problems might not have efficient algorithms for finding solutions, making them challenging to solve for large inputs.
NP-Pointer Network	A reinforcement learning-based algorithm for solving optimization problems like the Traveling Salesman Problem.

**Table 1.** *Cont.*

Term	Description
Gurobi	A commercial mathematical optimization solver used for linear programming, integer programming, mixed-integer programming, and more. It excels at solving large-scale and complex problems.
2-opt	A local search optimization algorithm employed to solve the TSP. It enhances the current solution by reversing two edges in the path, aiming for a shorter path.
OR-Tools	An open-source optimization library developed by Google. It is designed to address operations research problems like scheduling, routing, packing, and others.
Nearest Neighbor	The Nearest Neighbor algorithm is used to solve the TSP. It starts from an initial point and, at each step, selects the unvisited point closest to the current one, until all points have been visited.

## 2. Related Works

### 2.1. Collaborative Planning for Multi-UAV Missions

The advancement of computer and communication technologies has provided support for the scalability and intelligence of the IoT [11]. IoT devices connect to the internet and transmit data to the cloud for processing [12]. Intelligent sensors can be integrated with transportation infrastructure to achieve sustainable intelligent transportation systems [13]. With the increasing number of mobile users, IoT systems are required to handle massive amounts of data and user requests, generating a significant volume of data every day [14]. Efficient data transmission and processing capabilities are necessary. Cloud computing utilizes the internet and virtualization technologies to provide on-demand computing resources to users [15]. Wireless-powered mobile edge computing technology enhances the computing capabilities of mobile devices, compensating for limited battery capacity [16]. Partial computation offloading enables low-latency execution, benefiting latency-sensitive applications [17]. Various devices in the IoT are interconnected, connecting physical objects and supporting intelligent decision-making, device automation, and new services and applications [18]. The development of vehicular networks enables communication among vehicles, providing real-time traffic information and decision support to reduce road accidents and traffic congestion [19,20]. The development of smart vehicles offers a comfortable and safe travel environment for drivers and passengers [21]. Establishing a sustainable transportation system with zero emissions, energy efficiency, and diversified modes of transport will help reduce carbon emissions, save on fuel, and lower vehicle costs [22].

Renowned for their mobility and versatility, UAVs are widely utilized in various domains [23], providing essential computing, communication, and storage services to ground users [24]. UAVs play a crucial role in improving the safety and reliability of intelligent transportation systems [25]. However, UAV systems face challenges such as limited energy and flight restrictions [26], communication challenges, and large-scale data processing. Fog computing extends cloud computing infrastructure to the edge network [27,28], enhancing the collaborative capabilities of UAVs to handle complex environments and tasks, such as intelligent transportation [29], traffic prediction, and remote rescue operations. Network topology awareness identifies critical nodes, bottlenecks, and vulnerabilities, enhancing network robustness and performance. Shaikh et al. [30] proposed a topology structure and network monitoring algorithm based on Open Shortest Path First to improve network reliability, security, and performance. M. Laghate et al. [31] modeled the response mechanism of common communication protocols using Granger causality to infer directed data flows in group networks. Some researchers predict network objectives and behaviors by studying network evolution trends. H. Baek et al. [32] designed a Link-Situation Awareness and Control tactical data link for UAVs to ensure reliable UAV control and situational awareness. UAV mission processing involves predefining and coordinating the management of different numbers of UAVs, task types, and payloads to maintain a reasonable inter-UAV collaborative relationship. This processing covers multiple levels and aspects, including task allocation, trajectory planning, data link planning, and emergency response planning. Li et al. [33] addressed the problem of multi-UAV position optimization in Device-to-Device

networks and proposes a low-complexity GNN-based method to achieve optimal solutions. Tian et al. [34] introduced a three-step experience buffer depth deterministic policy gradient algorithm to enable rapid path planning for UAVs in urban environments. Li et al. [35] focused on data collection and flight trajectory optimization for UAVs to shorten mission completion time.

## 2.2. Application of Blockchain in Data Security for Multi-UAV Missions

Blockchain technology [36–38] has garnered significant attention due to its decentralized, anonymous, and immutable characteristics. Multi-UAV missions involve often involve a substantial amount of sensitive data, thereby underscoring the paramount significance of data security. The introduction of blockchain technology offers novel opportunities and solutions for the collaborative processing of UAV tasks. By incorporating blockchain technology into UAV networks, secure storage and sharing of UAV mission data can be accomplished, ensuring the integrity and reliability of data. Concurrently, blockchain technology can facilitate functionalities such as smart contracts, enabling trust management and automated collaboration among UAVs. This enhancement will further fortify the stability and security of UAV networks, thereby facilitating the proficient execution of multifaceted UAV collaborative tasks. Alladi et al. [39] provided an overview of the research progress on blockchain applications in UAVs and their various applications in UAV networks. Miao et al. [40] proposed a secure data sharing mechanism called BP2P-FL, which is based on privacy-preserving data providers in peer-to-peer federated learning, facilitating high-quality data sharing. Ma et al. [41] focused on delay-sensitive blockchain applications, analyzing the delay bounds of practical Byzantine fault-tolerant and HotStuff consensus algorithms using deterministic network calculus. This analysis offers valuable insights for research utilizing low-latency blockchain technology. Seid et al. [42] proposed an integrated blockchain and multi-agent deep reinforcement learning framework for computation offloading with EH in a multi-UAV-supported IoT network, where IoT devices obtain computing and energy resources from UAVs. Campos et al. [43] proposed a blockchain-based multi-UAV surveillance framework that enables UAV coordination and financial exchange between system users.

## 3. System Design

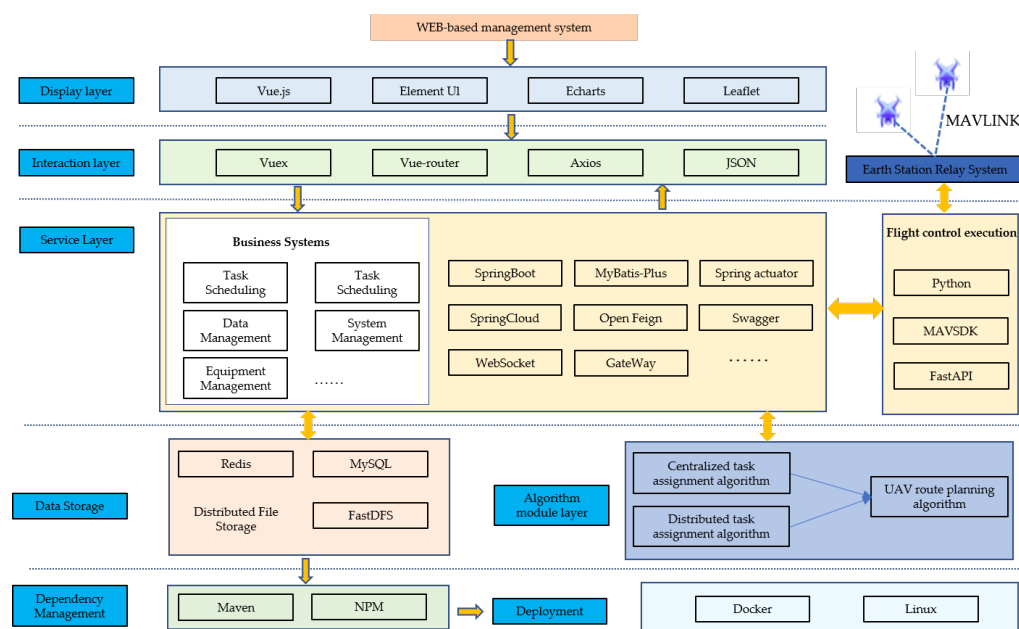
This section provides a detailed design of the system, including system architecture design, system function module design and main algorithm design, which will be followed by the detailed design of each part of the system.

### 3.1. System Framework Design

We employ a browser/server architecture, a decision that explicitly separates frontend and backend development, thereby capitalizing on the architecture's advantages. The primary merit of this architecture lies in its ability to decouple frontend and backend, reducing system coupling and allowing independent development at each level. Consequently, this enhances the system's maintainability and scalability as a whole. In terms of frontend development, we have chosen the Vue.js framework as the primary development tool. Vue.js, with its elegant design and user-friendly features, enhances the efficiency and flexibility of frontend development. Utilizing Vue.js' component-based approach, we divide the frontend interface into discrete components, fostering reusability and modularity, promoting team collaboration and parallel development. Furthermore, Vue.js' responsive design ensures real-time updates of the frontend interface based on user interactions, elevating the user experience. For backend development, we have combined the Spring Cloud, Spring Boot, and FastAPI frameworks. Spring Cloud equips the system with a plethora of tools and libraries for constructing a distributed architecture. This encompasses functionalities such as service discovery, load balancing, and distributed configuration, facilitating efficient service communication and management within a distributed environment. Spring Boot, as a framework for building microservices, expedites the establishment of standalone,



production-grade applications. Its streamlined configuration and automated functionalities curtail developers' workload, streamlining the development process. FastAPI, a modern Python-based web framework, is celebrated for its high performance and user-friendliness. Given the system's requirements for real-time perception and scheduling, FastAPI's rapid performance and asynchronous capabilities support real-time data processing and communication. The selection of these frameworks aims to address critical requirements such as high concurrency, availability, performance, reliability, and security within the system. Furthermore, the drone hardware scheduling layer employs MAVSDK for development, coupled with FastAPI to construct a remote communication ground station, facilitating real-time perception and scheduling of drones. Task scheduling and multi-drone route planning are integrated into the algorithmic module layer, divided into centralized and distributed task scheduling, along with route planning for drone mission services. In summary, the system architecture encompasses four principal components: frontend presentation and interaction layer, backend service layer, algorithmic scheduling layer, and data storage layer. Additionally, it encompasses dependency management and deployment strategies for development. The architecture is illustrated in Figure 1.



**Figure 1.** System architecture.

### (1) Frontend Presentation and Interaction Layer

The frontend presentation and interaction stratum shoulders the responsibility of presenting the system's frontend interface and facilitating user engagement. Within its scope, it encompasses pivotal functionalities encompassing login, registration, user administration, UAV oversight, task supervision, sensor oversight, and system administration. It is through this stratum that users can seamlessly engage with the system. The provision of a visual interface empowers task managers with real-time insights into UAV status and task advancement. This, in turn, facilitates timely task scheduling and informed decision-making. By offering an intuitive platform, the stratum affords users the opportunity to interact with the system, fostering efficient communication and enhanced operational control.

### (2) Backend Service Layer

The backend service layer is primarily responsible for handling user requests, including functionalities such as user login, registration, user management, task management, data management, and device management. The system also includes sub-functionalities such as real-time situational awareness, task scheduling, route planning, and task execution to achieve the goal of handling multiple real-time tasks for UAVs. To implement these

functionalities, the backend service will be designed in two parts: the main service component and the real-time situational awareness and scheduling module for UAV hardware. The main service component will be developed using technologies such as SpringCloud, SpringBoot, and MyBatis. The real-time situational awareness and scheduling module for UAV hardware will be developed using the FastAPI framework in Python combined with MAVSDK.

### (3) Algorithm Module Layer

The algorithm module layer is primarily responsible for system task scheduling and path planning. By combining various algorithms, it integrates the independent components of UAV task allocation and route planning into a unified whole, taking into account real-time tasks and UAV status to achieve optimal system scheduling. It consists of two main parts: a centralized and distributed hybrid UAV task allocation module and a path planning module based on reinforcement learning. During the task scheduling process, the system first uses a centralized algorithm to allocate tasks based on UAV resources and task information. If there are situations such as UAV disconnection or task abnormalities, the UAV will automatically use a distributed auction algorithm to reassign tasks, addressing the issue of centralized task allocation's inability to adapt to task changes. Once the task allocation is complete, the system will employ reinforcement learning methods to plan the paths for the UAVs, aiming to achieve the shortest service routes and minimize the energy consumption of the UAVs.

### (4) Data Storage Layer

During the operation of the system, a large amount of data are generated, and data storage and retrieval are essential components. This paper combines multiple technologies to achieve data storage and management. To accommodate different types of data storage and retrieval, MySQL, Redis, and FastDFS are used to store data and files. MySQL database is used to store user information, UAV information, task information, and other data. The Redis cache is used to store frequently accessed data, improving the query efficiency of the system. FastDFS is used to store large files such as perception images, providing high-performance and scalable file storage and management. It optimizes the performance of large file uploads and downloads, and ensures file backup and fault tolerance.

## 3.2. System Function Module Design

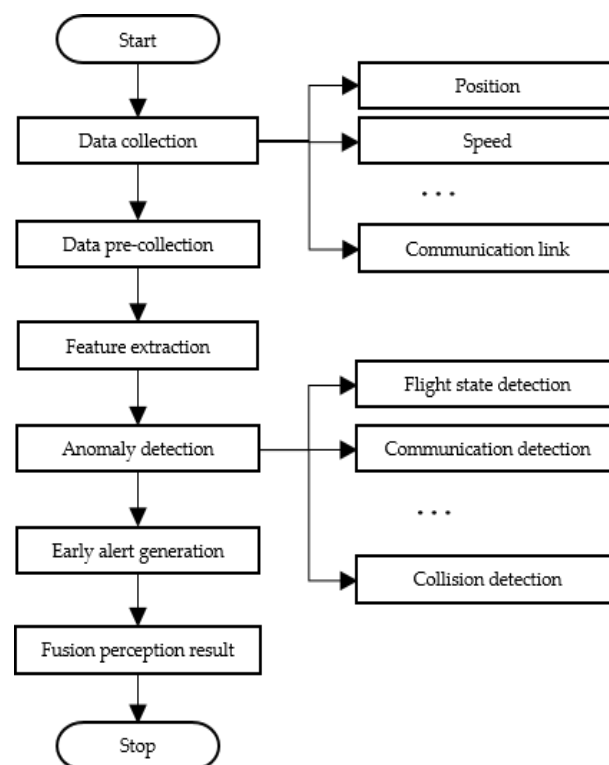
Building upon the system architecture design outlined in the previous section, this section will provide a detailed overview of each functional module within the system.

### 3.2.1. Hardware Situational Awareness and Scheduling Module Design

To ensure stability and adaptability in various environments, the hardware situational awareness and scheduling module has been designed. This module enhances the operational stability and safety of the UAV system by precisely controlling and providing real-time situational awareness.

The UAV control functionality is responsible for executing precise commands to the UAV, such as startup, flight, stop, and execution of specific tasks. The MAVSDK library is used to customize ground stations, enabling precise control of the UAV. This is particularly important in harsh environments, ensuring the stable operation of the UAV in complex conditions.

The situational awareness functionality is responsible for monitoring and early warnings. It collects and processes real-time flight data from the UAV, including information such as position, velocity, attitude, and environmental perception, to achieve real-time monitoring and early warnings in the UAV. Figure 2 illustrates the workflow of the module's perception and warning capabilities. By integrating perception and warning information, the module provides stability to the overall network situation, enabling timely UAV recalls to avoid accidents caused by UAV loss of control.



**Figure 2.** Network situation awareness and warning process diagram.

### 3.2.2. Task Allocation and Path Planning Module Design

The accomplishment of real-time scheduling and simultaneous processing of multiple UAV tasks necessitates meticulous algorithmic design, encompassing UAV task allocation algorithms and reinforcement-learning-based path-planning algorithms. The taxonomy of UAV task scheduling algorithms bifurcates into centralized and distributed paradigms. In the former, task allocation decisions are orchestrated by a central server, whereas the latter delegates task allocation autonomy to individual UAVs. To harness the merits of both paradigms, this study proposes a hybrid model amalgamating centralized and distributed task allocation mechanisms. Commencing with centralized task allocation, the system seamlessly transitions to distributed task reallocation whenever changes arise, thereby mitigating inter-UAV conflicts and augmenting system execution efficiency. Subsequent to curating task service enumerations for each UAV, path-planning algorithms facilitate meticulous route delineation. The comprehensive procedural delineation is visually represented in Figure 3.

#### (1) Design of Centralized Task Allocation Strategy

In the design of the centralized task scheduling algorithm in this system, a greedy strategy is employed as the basic approach. The advantage of the greedy strategy is its simplicity and ease of implementation, which allows for quick task allocation results in a centralized system.

The solution to this problem is obtained using the greedy strategy, where the idea is to prioritize the scheduling of tasks with the highest service priority, while satisfying the energy requirements. The pseudocode for this approach is shown in Algorithm 1.

During the task execution process, the energy status of the UAV is a critical factor to consider. It is important to prioritize the energy status of UAVs to avoid task delays or the inability to complete tasks. The UAV collection is sorted in descending order of remaining energy, ensuring that UAVs with higher energy levels are selected first. Then, the task collection is sorted in descending order of priority, and the collection is iterated to find the UAV with the maximum remaining energy that meets or exceeds the energy requirements of the task. If a suitable UAV is found, it is assigned the task and its remaining energy is



updated. If no suitable UAV is found, options such as increasing the number of UAVs or recharging should be considered. Finally, the task-to-UAV assignment plan is returned.

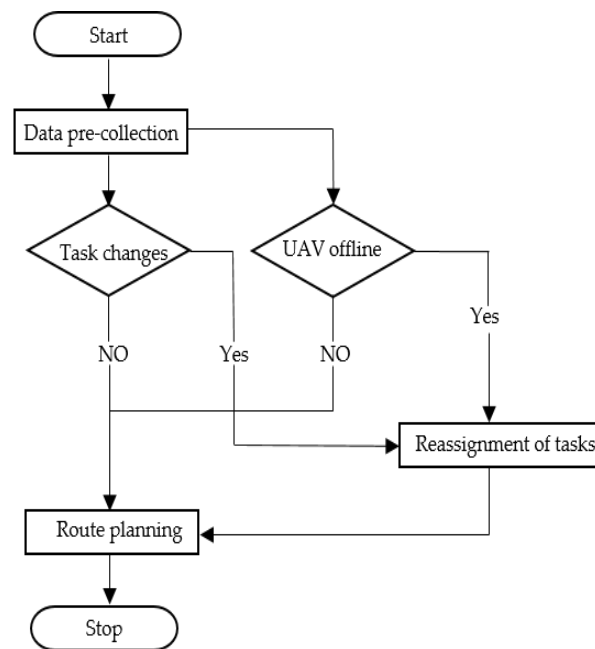


Figure 3. Task assignment and route planning process plowchart.

---

**Algorithm 1** The Greedy Algorithm-based UAV Task Scheduling Algorithm

---

**input:**  $T$ , Set of tasks;  $D$ , set of UAVs;  $C_t$ , Energy consumption of task;  $E_d$ , Remaining energy of UAV  $d$ ;  $p_t$ , Service priority of task  $t$ ;

**output:**  $\pi$ , Solution for UAV task allocation;

initialize:  $\pi = \theta$ ;

Sort the tasks in descending order of service priority;

**for all**  $t \in T$  **do**

Find the UAV  $d^*$  with remaining energy greater than or equal to, such that the remaining energy of  $d^*$  is maximized;

**if** The UAV  $d^*$  has been found **then**

$\pi(t) = d^*$ ;

$E_{d^*} \leftarrow E_{d^*} - C_t$ ;

**else**

**return** No solution found.

**end if**

**end for**

**return**  $\pi$ .

---

## (2) Design of Distributed Task Redistribution Strategy

In order to enhance task scheduling efficiency, mitigate communication overhead, minimize processing delays, and augment system flexibility, an edge-deployed distributed task redistribution algorithm is implemented. This algorithm facilitates the dynamic reallocation of tasks among UAVs, leveraging their prevailing states. This proves especially beneficial in exigent scenarios, ultimately bolstering task completion efficiency. The edge-centric dynamic task redistribution algorithm takes several pivotal factors into meticulous consideration, encompassing device resource constraints, network bandwidth limitations, task attributes, and system dynamics. Fundamentally rooted in auction strategies, the algorithm treats the task assignment conundrum as an auction scenario, where UAVs function as bidders and tasks embody the auction items. UAVs vie for tasks contingent upon their respective statuses and task requisites, culminating in the assignment of a task to the

UAV that proffers the most economical bidding cost. This auction-based algorithm confers noteworthy merits in the realm of decentralized decision-making and task allocation, thereby diminishing the dependence on central servers. During the operational lifecycle of the system, the redistribution of tasks is dynamically fine-tuned in consonance with real-time conditions, endowing the auction algorithm with the agility to seamlessly adapt to fluctuations in real-world scenarios. Central to the algorithm's efficacy is the judicious selection of the UAV presenting the most economical bidding proposition for task execution. This judicious curation ensures the seamless and robust redistribution of tasks, thereby optimizing the efficiency of task completion and the utilization of UAV resources. A formal representation of the pseudocode governing task redistribution, hinging on the auction strategy, is delineated in Algorithm 2.

---

**Algorithm 2** Task Allocation Process for UAVs based on Auction Algorithm

---

**input:** Set of UAVS  $U$ , Set of tasks  $T$ ;  
**output:** Solution for UAV task allocation;  
 initialize: Initialize the task assignment collection  $unassigned\_tasks \leftarrow T$ ;  
**while**  $unassigned\_tasks \neq \emptyset$  **do**  
   Initialize the minimum cost:  $min\_cost \leftarrow \infty$ ;  
   Initialize the UAV corresponding to the minimum cost  $min\_UAV \leftarrow None$ ;  
   **for**  $UAV \in U$  **do**  
     **for**  $task \in unassigned\_tasks$  **do**  
       **if**  $task.energy\_requirement > UAV.battery$  **do**  
         Continue with the next iteration;  
       **end if**  
       Calculate the cost:  $cost \leftarrow calculate\_cost(UAV.task)$ ;  
       **if**  $cost < min\_cost$  **then**  
         Update the minimum cost:  $min\_cost \leftarrow cost$ ;  
         Update the UAV corresponding to the minimum cost:  $min\_UAV \leftarrow UAV$ ;  
         Update the task corresponding to the minimum cost:  $min\_task \leftarrow task$   
       **end if**  
     **end for**  
   **end for**  
   **if**  $min\_UAV \neq None$  and  $min\_task \neq None$  **then**  
      $min\_UAV$  Complete the task  $min\_task$ ;  
     Remove the task  $min\_task$  from  $unassigned\_tasks$ ;  
   **else**  
     End the loop.  
   **end if**  
**end while**  
**return** Allocation result.

---

In the above pseudocode, the function  $calculate\_cost()$  calculates the cost by considering the distance, energy, and priority between the UAV and the task. It combines the calculated distance cost, energy cost, and priority cost to obtain the total cost, forming a comprehensive cost calculation method.

### (3) Design of Reinforcement Learning-based Path Planning Algorithm

The aforementioned task scheduling algorithm has successfully achieved efficient task allocation. However, practical applications also necessitate the consideration of UAV flight path planning. During flights, UAVs must select suitable paths based on task requirements to ensure optimal task completion. This path planning conundrum can be conceptualized as the Traveling Salesman Problem (TSP). Given its NP-hard complexity, conventional solving techniques often demand substantial computational resources and struggle to address extensive-scale scenarios. Consequently, our system adopts a reinforcement-learning-based path-planning approach known as the Pointer Network. In contrast to conventional TSP-solving methods, the Pointer Network offers a novel solution that excels at addressing large-scale TSP challenges. This is attributed to its quicker computation speed and en-

hanced accuracy. By leveraging the strengths of the Pointer Network algorithm, our system overcomes the limitations of traditional techniques and efficiently handles complex flight path-planning for UAVs.

The Pointer Network is mainly applied to Sequence-to-Sequence (Seq2Seq) learning problems, aiming to address the core issue in sequence generation: how to select elements from the input sequence to make the output sequence more accurate. Its core idea is to map each element of the input sequence to the corresponding position in the output sequence, resulting in the final output sequence. The Pointer Network consists of two main components: the Encoder and the Decoder, as shown in Figure 4. In the Encoder part, the input sequence is transformed into a high-dimensional representation in a mapping process. The Decoder part generates the output sequence step-by-step based on the information in this representation. Unlike traditional Seq2Seq models, the Pointer Network introduces a pointer mechanism in the Decoder, allowing the selection of elements from the input sequence in the output sequence. Specifically, the Decoder learns to generate a probability distribution for each position, corresponding to the elements of the input sequence. The Decoder then utilizes this probability distribution to determine which position's element to select in the output sequence. This pointer mechanism enables the model to directly "point" to the elements that need to be output from the input sequence, rather than only generating words from a fixed vocabulary to construct the output sequence. Through this approach, the Pointer Network can effectively solve optimization problems such as the TSP, which requires finding the shortest path to visit a set of locations.

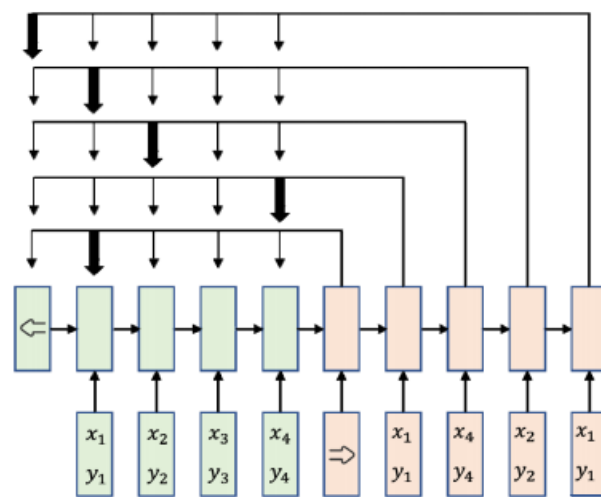


Figure 4. Pointer Network structure.

The Encoder part employs a Bidirectional Recurrent Neural Network (BiRNN) to transform the input sequence  $x = (x_1, x_2, \dots, x_n)$  into an encoding matrix  $H = (h_1, h_2, \dots, h_n)$ , where  $h_i$  represents the encoding of the  $i$ -th element in the input sequence. Specifically, the BiRNN consists of two Recurrent Neural Networks (RNNs) that encode the input sequence from left to right and from right to left, respectively. The outputs from these two directions are then concatenated to obtain the final encoding result.

The Decoder part consists of an RNN that takes as input the previous output and the encoding matrix from the Encoder. At each timestep, the Decoder calculates an attention distribution, which indicates which parts of the input sequence should be attended to at the current position. This attention distribution is used to compute a weighted average, representing which element from the input sequence should be chosen as the next element in the output sequence.

In more detail, let  $y_t$  denote the output at timestep  $t$ ,  $s_t$  denote the state vector at timestep  $t$ ,  $c$  denote the context vector, and  $h_t$  denote the input representation. The computation of the Decoder can be expressed as follows:

$$s_t = f(y_{t-1}, s_{t-1}, c) \quad (1)$$

$$P(y_t = i | y_1, \dots, y_{t-1}, c) = g(h_t, s_t) \quad (2)$$

In the computation,  $f$  and  $g$  represent Multi-Layer Perceptron models. The term  $P(y_t = i | y_1, \dots, y_{t-1}, c)$  represents the probability of selecting the  $i$ -th input. The variable  $h_t$  represents the input representation, which can be obtained by using the last state vector of the Encoder, specifically  $h_t = h_{L+1}$ . In this process, the Decoder takes as input the sequence  $y_1, \dots, y_{t-1}$ , and the context vector  $c$ , and predicts the next output  $y_t$ . This process continues until all input sequences have been predicted.

In the Pointer Network, a Pointer Mechanism is introduced to perform pointer operations on the input sequence, mapping the probability distribution outputted by the Decoder to the input sequence. Specifically, given the current state  $s_t$  of the Decoder, the position of the pointer at timestep  $t$  is calculated as follows:

$$p_t = \sum_{i=1}^{T_x} \alpha_{ti} h_i \quad (3)$$

where  $\alpha_{ti}$  represents the probability of input  $i$  corresponding to the output  $t$ . It can be expressed as:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \quad (4)$$

where  $e_{ti}$  represents the correlation between input  $i$  and output  $t$ , and can be expressed as:

$$e_{ti} = v^T \tanh(W_h s_t + W_e h_i) \quad (5)$$

where  $W_h$ ,  $W_e$ , and  $v$  are weight matrices, and  $\tanh$  represents the hyperbolic tangent function.

Through the Pointer Mechanism, the Decoder can predict the next output based on the previous output and the context vector, and can also perform pointer operations on the input sequence for improved performance. When applied to TSP problem solving, the pseudocode for this model is shown in Algorithm 3.

In the above design, the problem is solved through two stages: Encoder and Decoder. In the Encoder stage, the algorithm extracts features from the input set of points and feeds each point's feature vector into the Encoder, obtaining the Encoder's output and state. The purpose of this stage is to encode the feature information of the input points into the hidden state of the Encoder for subsequent path planning.

In the Decoder stage, the path is generated iteratively by looping. First, the Decoder's state is initialized as the last hidden state of the Encoder. Through the Pointer Mechanism, the Decoder's state, previous output, and the Encoder's output set are input into the model. The Pointer Mechanism calculates weight scores for each point based on the current state and the history of outputs, and normalizes the scores into a probability distribution. Then, the algorithm samples from the probability distribution to obtain the index of the next point to be visited and adds it to the path. The Decoder's state is then updated and the output is calculated. This loop iteration continues until the path length reaches the specified size of the point set.

Finally, the distance from the last point to the starting point is calculated and added to the path length, resulting in the final shortest path distance. The model returns the shortest path distance and the order of cities visited, which in this system corresponds to the order of service for the UAVs.

**Algorithm 3** Pointer Network solves the TSP problem

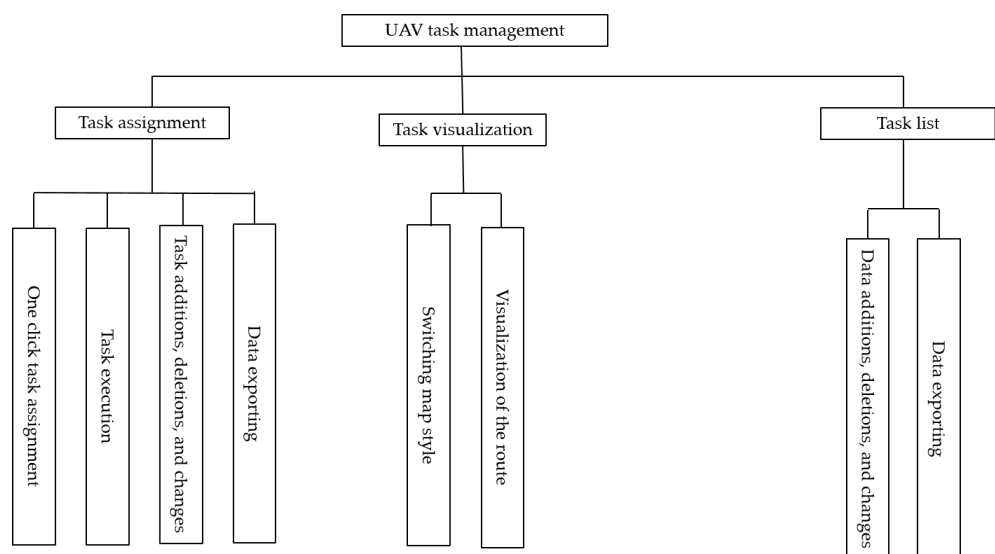
---

**input:** TSP problem input point set  $P$  with point set size  $n$ ;  
**output:** Shortest path distance  $d$ , shortest path  $path$ ;  
 Feature extraction is performed on the point set  $P$  to obtain the feature vector  $f_i, i = 1, \dots, n$ ;  
 Initialize the Encoder's state  $h_0$ ;  
**ForEach**  $i \leftarrow 1$  to  $n$   
   Input the feature vector  $f_i$  and the state  $h_{i-1}$  of the last Encoder into the Encoder;  
   Compute the Encoder's output and state  $o_i, h_i = \text{Encoder}(f_i, h_{i-1})$ ;  
**end**  
 Initialize the Decoder's state  $s = h$ ;  
 Initialize path length  $L=0$  and  $path=s_0$ ;  
**while**  $path < n$  **do**  
   Input the Decoder's state  $s_{t-1}$  and last output  $y_{t-1}$  as well as the Encoder's output set  $O = o_1, o_2, \dots, o_n$ , into the Pointer Mechanism;  
   Calculate the weight score  $p_i = \text{score}(s_{t-1}, o_i)$  for each point, where *score* is the scoring function;  
   Normalizing the weight scores with the *Softmax* function yields a probability distribution  $p = p_1, p_2, \dots, p_n$ ;  
   Sample from the probability distribution  $p$  to obtain the index  $t$  of the point to be visited next;  
   Calculate the current path length  $L = L + d(s_{t-1}, s_t)$  and add point  $s_t$  to the path  $Path = path \cup s_t$ ;  
   Update the Decoder's state  $s_t$  and compute the output  $y_t$ ;  
**end while**  
 Calculate the distance from the last point to the starting point  $L = L + d(s_n, s_0)$ ;  
**return** the shortest path distance  $d = L$  and the shortest path  $path$ .

---

**3.2.3. UAV Task Management Design**

The UAV task management module serves as the frontend interface for task management, allowing operators to oversee and control tasks. It interfaces with both the task assignment and route planning modules to acquire algorithm-generated outcomes. Furthermore, it collaborates with the hardware situational awareness and scheduling module to allocate and schedule tasks. Additionally, it establishes communication with the backend management system to facilitate task monitoring and administration. The structural layout of this module is illustrated in Figure 5.



**Figure 5.** Task scheduling management module diagram.

The task allocation module is the core component of UAV task management. It displays all pending task information and provides task assignment and execution functionality.

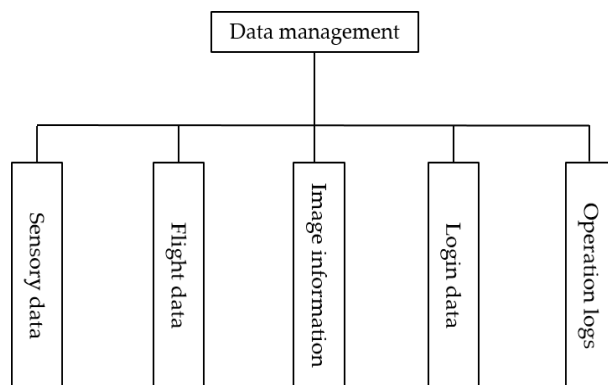


Through this interface, administrators can select specific tasks and the system will use the centralized task allocation functionality of the Task Allocation and Route Planning module to assign the tasks. Once task allocation is complete, administrators can trigger task execution for the UAVs from this page, and the ground station will execute the task commands.

The task visualization feature provides a visual representation of the UAV task execution routes and task completion status. Through maps and route diagrams, the current position of each UAV, task execution routes, task distribution, and task completion status can be observed, enabling real-time monitoring of task execution for the operators. The Task List functionality displays all task information, including task execution time, assigned UAVs, and task status. Administrators can query all tasks from this page for statistical analysis and monitoring of task execution. It also supports task data management operations such as add, delete, modify, and search, as well as Excel data export, fulfilling the requirements for task data management.

### 3.2.4. UAV Perception Data Management Design

A large amount of data are generated during the operation of the system, and the Data Management module is responsible for collecting and managing these data. These data include various types of sensor data collected by the UAVs, such as environmental information and target detection data. The Data Management Module ensures that these data can be uploaded to the system in real time, and provides operators with the ability to query and manage the data. A schematic diagram of the Data Management Module is shown in Figure 6.

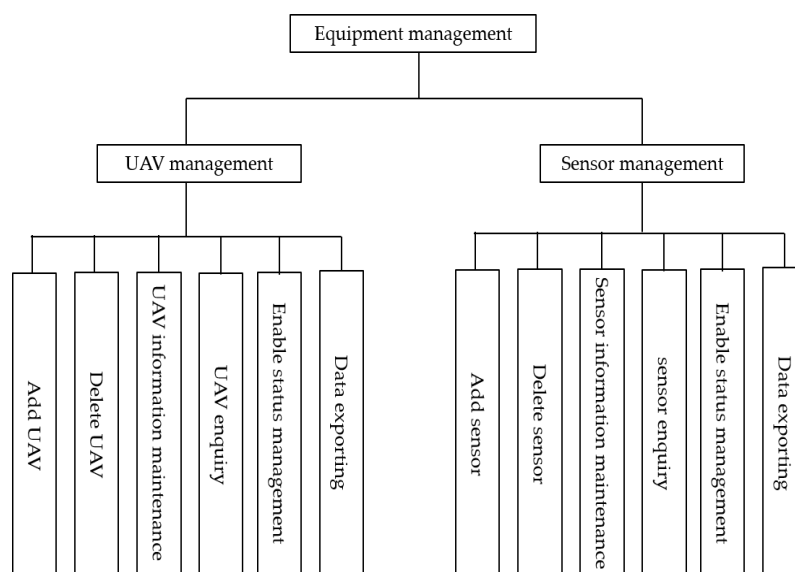


**Figure 6.** Data management module diagram.

The Data Management module interacts with the Ground Control Station to provide data collection and storage services. It manages various types of data and utilizes MySQL for persistent storage of record-based data. For image- and file-based data, a distributed storage system like FastDFS is used. Additionally, the data management module also provides data querying and export functions, allowing operators to easily search and analyze the data.

### 3.2.5. UAV Equipment Management Design

UAV equipment management is responsible for managing all the UAVs and sensors that are stored in the inventory. This includes operations such as adding, deleting, querying, and modifying UAV devices, as well as adding, deleting, querying, and modifying sensors. The structure diagram of the UAV equipment management module is shown in Figure 7, which enables centralized management for easy maintenance and administration of the devices, ensuring the efficient operation and accurate execution of the UAV system.



**Figure 7.** User management module diagram.

The UAV management module assumes the critical responsibility of monitoring and scheduling all UAV devices registered in the system. Personnel have the authority to determine a device's participation in task execution by configuring its enablement status. This status holds paramount importance within the task scheduling module. Specifically, task assignment solely considers UAVs marked as enabled, while those lacking this status remain excluded from consideration. Similarly, the sensor management module handles the administration of all sensor devices within the system, establishing their association with corresponding UAV devices. Personnel-set enablement status once again governs the initiation of sensor operation. Enabled sensors have their data comprehensively recorded and managed by the data management module. In contrast, data from disabled sensors remains unrecorded and untouched. Through the interplay of these modules, the system effectively governs the UAVs and their associated sensors. This orchestrated synergy ensures operational coherence, allowing for efficient management of tasks and sensor data.

#### 4. System Implementation and Performance Analysis

This section delves into the practical realization of the system's functionalities, building upon the foundation laid out by the functional design. The focal points encompass the hardware situational awareness and scheduling module, the task assignment and route planning module, data management, and user management functionalities. The subsequent sections provide a comprehensive explanation and demonstration of the implementation process for each of these pivotal functionalities.

##### 4.1. Implementation of the Hardware Situational Awareness and Scheduling Module

The real-time hardware situational awareness and scheduling module perceive various data from the system's network situation. It integrates warning information and provides corresponding scheduling plans in case of adverse network situations. MAVSDK and FastAPI are used for development to achieve UAV scheduling control and real-time situational awareness, ensuring the timeliness and accuracy of system situational awareness. The UAV scheduling control functionality is implemented using MAVSDK. It provides a high-level encapsulation of the MAVLink protocol, allowing simple API calls to control UAV actions such as takeoff, flight, landing, and advanced operations like route cruising. In this system, a ground station control platform is developed using Python-based MAVSDK.

The hardware situational awareness module utilizes UAV sensors to collect environmental data and transmits these data in real-time to the ground control station. MAVSDK's data streaming service is used to transmit UAV sensor data to the ground control station,

and FastAPI is utilized to develop the service invocation endpoint, which receives and processes these data to generate real-time situational information and warning messages for the UAVs. The situational awareness and warning information is displayed on the frontend page, which also integrates UAV scheduling and control functionalities. The page layout is depicted in Figure 8.

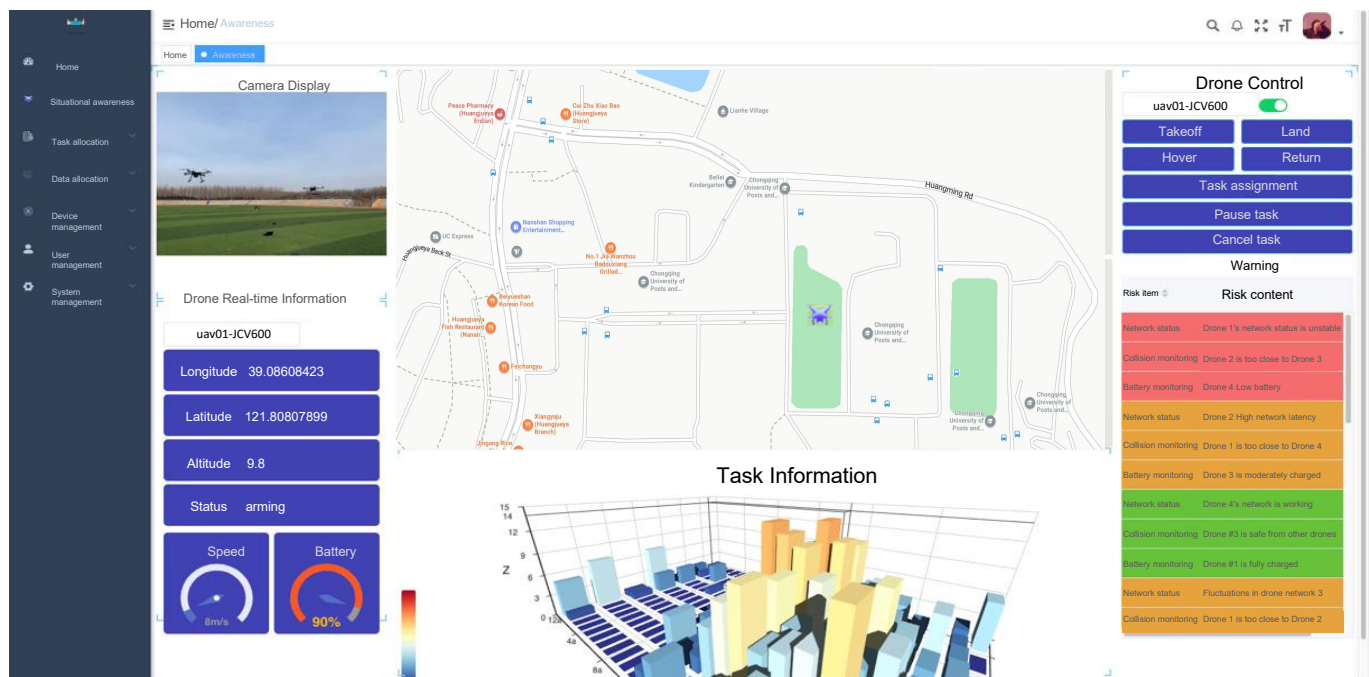


Figure 8. Situation monitor and control.

The left section displays the onboard camera feed and UAV's attitude, altitude, speed, position, and other information. The middle section uses a map API to display real-time locations and task hotspots. The right section contains the control module and situational awareness warning information, enabling real-time maneuvering control of the UAVs. Situational awareness and warning information, such as collision risks between UAVs based on their real-time positions, stability status of communication links with the ground station, and remaining battery capacity, are displayed based on server-side processing.

By integrating network situational awareness and warning results, the module promptly provides warning information when multiple UAVs are in unfavorable conditions such as disconnection or poor communication links. This alerts the personnel to take timely actions, such as returning the UAVs or terminating ongoing tasks. An example warning is shown in Figure 9.

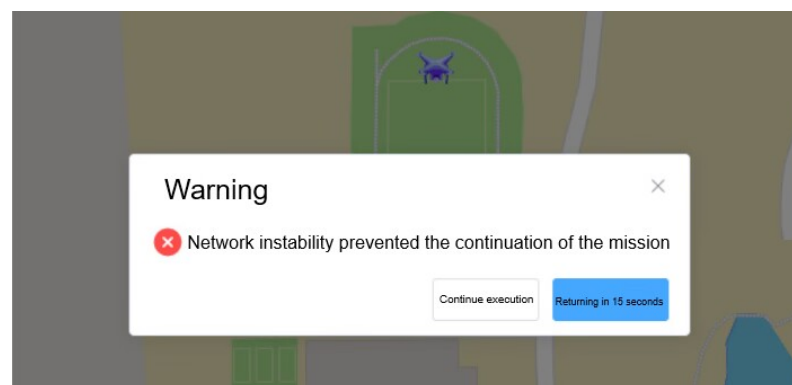
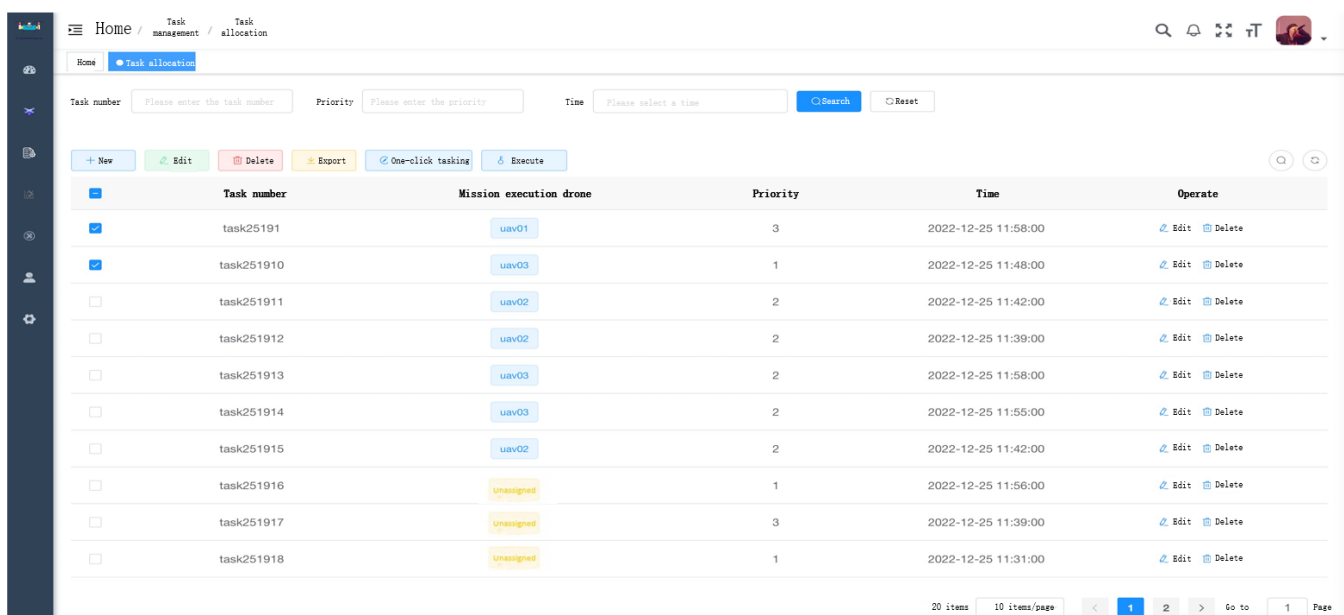


Figure 9. Situation alert result.

#### 4.2. Implementation of UAV Mission Management

The implementation of the mission scheduling and management feature takes charge of efficiently scheduling and overseeing tasks uploaded by users or administrators. Its principal role revolves around dispatching precise task instructions to individual UAVs and guiding them in the seamless execution of their designated tasks. To achieve this, the module operates in close concert with the algorithm scheduling layer framework, which is realized using the Spring Cloud microservice architecture. The module's communication with the algorithm service is facilitated through well-defined interfaces, enabling the retrieval of scheduling outcomes.

When a user or administrator selects their desired tasks and triggers the 'One-click tasking' button, the module promptly engages in centralized scheduling by leveraging the outcomes derived from the algorithm scheduling layer. This orchestration process ensures optimized allocation of tasks. As depicted in Figure 10, the module subsequently displays a diverse array of task allocation statuses once the scheduling process concludes. Building upon these scheduling results, the module meticulously plans optimal routes for each UAV. This route planning takes into account the UAV's task load and geographic location, thereby guaranteeing that each UAV is equipped with the most efficient task execution path. With this groundwork laid, the 'Execute Task' button comes into play. By selecting this option, the module effectively dispatches tasks to individual UAVs in accordance with the allocation and route planning outcomes stipulated by the algorithm scheduling layer.



Task number	Mission execution drone	Priority	Time	Operate
task25191	uav01	3	2022-12-25 11:58:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251910	uav03	1	2022-12-25 11:48:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251911	uav02	2	2022-12-25 11:42:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251912	uav02	2	2022-12-25 11:39:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251913	uav03	2	2022-12-25 11:58:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251914	uav03	2	2022-12-25 11:55:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251915	uav02	2	2022-12-25 11:42:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251916	Unassigned	1	2022-12-25 11:56:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251917	Unassigned	3	2022-12-25 11:39:00	<a href="#">Edit</a> <a href="#">Delete</a>
task251918	Unassigned	1	2022-12-25 11:31:00	<a href="#">Edit</a> <a href="#">Delete</a>

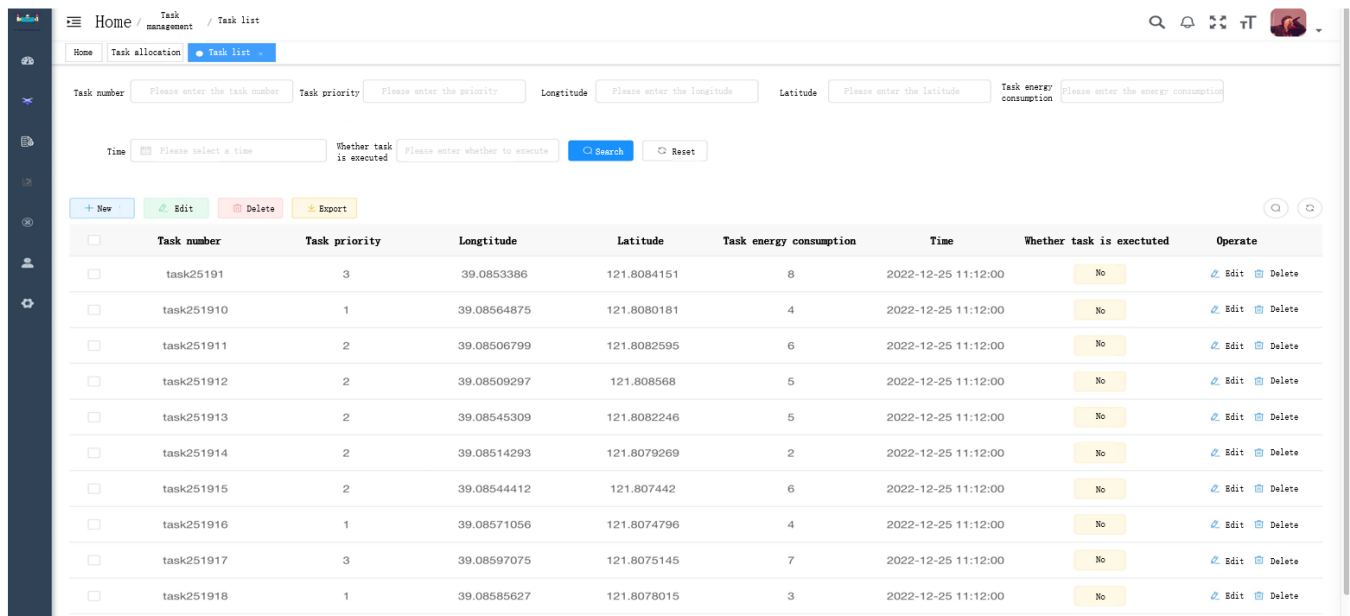
Figure 10. Task assignment management page.

The system provides a task list page where all tasks can be retrieved and viewed with detailed information. Additionally, the personnel can edit and delete tasks through this system. The user interface for this feature is depicted in Figure 11.

#### 4.3. Implementation of UAV Perception Data Management

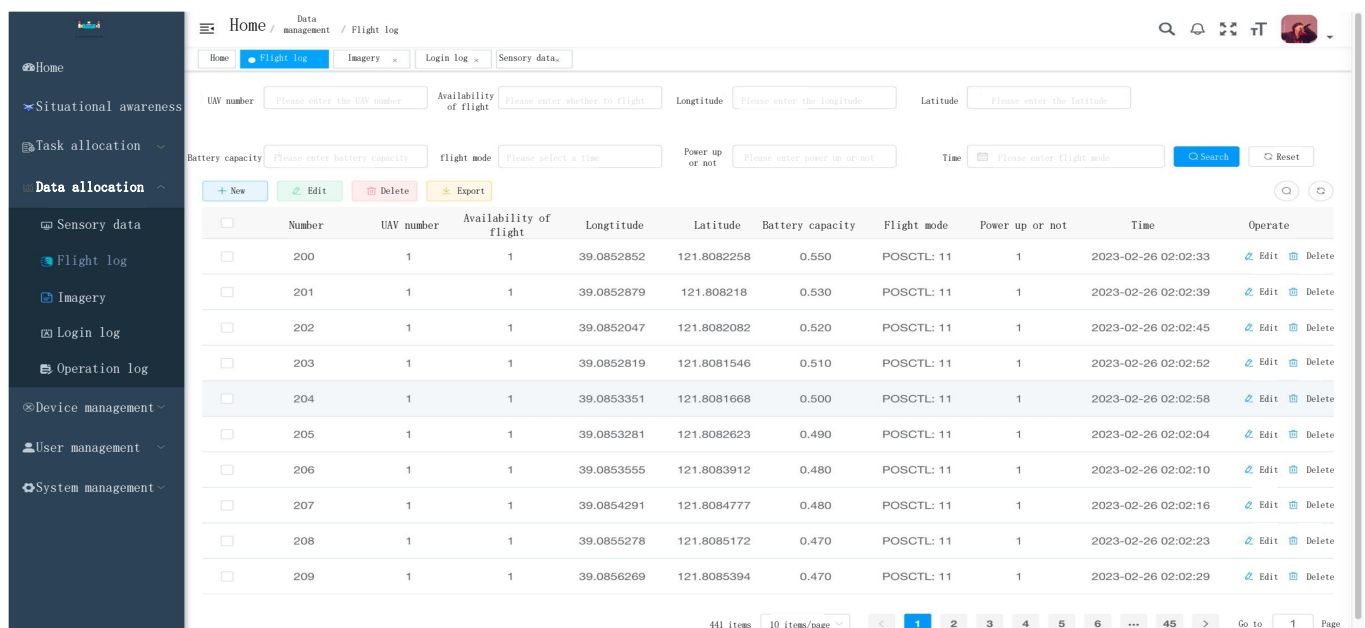
The perception data management microservice is built to communicate with the MAVSDK ground station using FastAPI. It facilitates real-time retrieval of UAV's current status, operational logs, and image files, which are then recorded accordingly. The perceived data can be categorized into two main types: log records and image file storage, with support for Excel data import and export. MySQL is used to record log-type data, capturing the UAV's operational status, detailed task execution records, and sensor data for comprehensive analysis and monitoring of the UAV's performance. Additionally, the sys-

tem stores records of personnel's actions as part of the system's data. As an example, the implementation of flight data are shown in Figure 12.



Task number	Task priority	Longitude	Latitude	Task energy consumption	Time	Whether task is executed	Operate
task25191	3	39.0853386	121.8084151	8	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251910	1	39.08564875	121.8080181	4	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251911	2	39.08506799	121.8082595	6	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251912	2	39.08509297	121.808568	5	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251913	2	39.08545309	121.8082246	5	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251914	2	39.08514293	121.8079269	2	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251915	2	39.08544412	121.807442	6	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251916	1	39.08571056	121.8074796	4	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251917	3	39.08597075	121.8075145	7	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>
task251918	1	39.08585627	121.8078015	3	2022-12-25 11:12:00	No	<a href="#">Edit</a> <a href="#">Delete</a>

Figure 11. Task list page.



Number	UAV number	Availability of flight	Longitude	Latitude	Battery capacity	Flight mode	Power up or not	Time	Operate
200	1	1	39.0852852	121.8082258	0.550	POSCTL: 11	1	2023-02-26 02:02:33	<a href="#">Edit</a> <a href="#">Delete</a>
201	1	1	39.0852879	121.808218	0.530	POSCTL: 11	1	2023-02-26 02:02:39	<a href="#">Edit</a> <a href="#">Delete</a>
202	1	1	39.0852047	121.8082082	0.520	POSCTL: 11	1	2023-02-26 02:02:45	<a href="#">Edit</a> <a href="#">Delete</a>
203	1	1	39.0852819	121.8081546	0.510	POSCTL: 11	1	2023-02-26 02:02:52	<a href="#">Edit</a> <a href="#">Delete</a>
204	1	1	39.0853351	121.8081668	0.500	POSCTL: 11	1	2023-02-26 02:02:58	<a href="#">Edit</a> <a href="#">Delete</a>
205	1	1	39.0853281	121.8082623	0.490	POSCTL: 11	1	2023-02-26 02:02:04	<a href="#">Edit</a> <a href="#">Delete</a>
206	1	1	39.0853555	121.8083912	0.480	POSCTL: 11	1	2023-02-26 02:02:10	<a href="#">Edit</a> <a href="#">Delete</a>
207	1	1	39.0854291	121.8084777	0.480	POSCTL: 11	1	2023-02-26 02:02:16	<a href="#">Edit</a> <a href="#">Delete</a>
208	1	1	39.0855278	121.8085172	0.470	POSCTL: 11	1	2023-02-26 02:02:23	<a href="#">Edit</a> <a href="#">Delete</a>
209	1	1	39.0856269	121.8085394	0.470	POSCTL: 11	1	2023-02-26 02:02:29	<a href="#">Edit</a> <a href="#">Delete</a>

Figure 12. Log type data management page.

We harness the capabilities of the open-source FastDFS distributed file system to effectively store the collected image files. This strategic choice not only ensures efficient storage, but also bolsters reliability due to its inherently distributed architecture. By leveraging FastDFS's API interface, we seamlessly execute a spectrum of operations including uploading, downloading, and deletion of files. Additionally, this framework supports real-time previews of the files through their designated URLs, enhancing user accessibility to these resources. The implementation result is depicted in Figure 13.



File name	<input type="text" value="Please enter the file name"/>	File path	<input type="text" value="Please enter the file path"/>	Time	<input type="text" value="Please select a time"/>	<input type="button" value="Search"/>	<input type="button" value="Reset"/>
<input type="button" value="+ New"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Export"/>							
<input type="checkbox"/>							
<input type="checkbox"/>	21	0h5h7mQAnpyATL8QAAJqLJzz1_...	http://210.30.97.238:8888/group1/...	2023-03-02 13:03:22	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	22	0h5h7mQAnq6AABvhAAJqLJzz1_...	http://210.30.97.238:8888/group1/...	2023-03-02 13:03:40	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	28	0h5h7mQAvuiAMGOIAAGuagn_0Bl...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:13	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	29	0h5h7mQAw0KAOu_UAAypFYJzO...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:42	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	30	0h5h7mQAwUSAKWgbAAJqLJzz1...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:16	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	31	0h5h7mQAwVOAI-uvAAMGCgK0O...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:32	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	32	0h5h7mQAwX-AEuaiAAGuagn_0Bl...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:15	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	33	0h5h7mQAxUJAzhcAAJqLJzz1_w...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:20	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	34	0h5h7mQAxVKASO9qAAGuagn_0...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:34	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<input type="checkbox"/>	35	0h5h7mQAxeiAGh32AAA1wnbrHz0...	http://210.30.97.238:8888/group1/...	2023-03-02 15:03:06	<a href="#">Download</a>	<a href="#">Edit</a>	<a href="#">Delete</a>

**Figure 13.** Image data management page.

#### 4.4. User Management Implementation

The user management module in this system plays a pivotal role in administering user-related functions, encompassing tasks such as user login, logout, user information management, and role administration. The initial login process involves validating the provided username and password against the records stored in the MySQL database. Following a successful login, a token is generated and conveyed to the frontend for storage, concurrently being stored in Redis. Subsequent requests originating from the frontend are subject to validation, with the token being decoded through the gateway. Successful validation ensures that users are directed to the appropriate pages. After user login, different roles will have different permissions. For example, after an administrator logs in, they can access the user information management page as depicted in Figure 14.

<a href="#">Home</a> <a href="#">Role manager</a> <a href="#">User manager</a>						
User name	<input type="text" value="Please enter the user name"/>	Phone number	<input type="text" value="Please enter phone number"/>	statuses	<input type="text" value="User statuses"/>	
Creation time	<input type="text" value="Start time"/>	<input type="text" value="End time"/>	<input type="button" value="Search"/>	<input type="button" value="Reset"/>		
<input type="button" value="+ New"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Import"/> <input type="button" value="Export"/>						
<input type="checkbox"/>						
<input type="checkbox"/>	1	admin	super	15712103456	<input checked="" type="checkbox"/>	2023-01-15 13:15:06
<input type="checkbox"/>	2	uavT	uavT	15612345678	<input checked="" type="checkbox"/>	2023-01-15 13:15:06 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	100	newname	name		<input checked="" type="checkbox"/>	2023-01-16 12:19:31 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	101	jia	jia		<input checked="" type="checkbox"/>	2023-02-18 06:04:21 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	102	fly	fly	15712345690	<input checked="" type="checkbox"/>	2023-02-18 08:40:07 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	103	sk	sk		<input checked="" type="checkbox"/>	2023-02-23 06:47:31 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	104	gao	gao		<input checked="" type="checkbox"/>	2023-02-28 06:37:28 <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	105	liq	liq		<input checked="" type="checkbox"/>	2023-02-28 10:32:54 <a href="#">Edit</a> <a href="#">Delete</a>

**Figure 14.** Role management page.

#### 4.5. Performance Analysis

Optimizing UAV flight trajectories is pivotal in maximizing the utilization of UAV power, thus significantly impacting system performance. The objective is to ensure that the UAV efficiently covers all areas while minimizing the route length. In this system, the optimization of UAV routes is accomplished using the Pointer Network.

For the performance comparison experiments, a deep learning platform built upon PyTorch 1.13.1 was employed, with Python as the programming language. The experiments were conducted on the Ubuntu 20.04 operating system, utilizing the Nvidia GeForce RTX 3090 Founders Edition GPU for model training. The model employed is a single-layer LSTM model comprising 512 hidden units. Training was conducted through stochastic gradient descent with a learning rate of 1.0 and a batch size of 128. Model weights were initialized via a uniform random distribution ranging from  $-0.08$  to  $0.08$ . For optimal TSP gap determination, the Gurobi solver was employed. This commercial mathematical programming solver is adept at delivering optimal solutions for TSP problems involving up to 100 nodes. Traditional solving tools and algorithms, including OR-Tools, 2-opt, and the Nearest Neighbor algorithm, were also incorporated for comparison. The experiments were executed on an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz.

Three different datasets were used in the experiments: 20-TSP, 50-TSP, representing TSP problems with 20, 50, and 100 nodes, respectively. For all node experiments, the node coordinates were represented as  $(x_i, y_i)$ , randomly distributed within a square of  $[0,1] \times [0,1]$ . The same data distribution was used for both training and testing stages. Based on the above settings, the results of the various models after the experiments are shown in Table 2.

**Table 2.** Performance comparison of various model.

Mode	20-TSP			50-TSP			100-TSP		
	Cost	Interval	Time	Cost	Interval	Time	Cost	Interval	Time
Gurobi	3.83	0.00%	7 s	5.69	0.00%	2 min	7.76	0.00%	17 min
2-opt	3.95	3.13%	1 s	6.11	7.38%	7 s	8.5	9.53%	33 s
OR-Tools	3.86	0.94%	1 min	5.85	2.81%	5 min	8.06	3.87%	23 min
Nearest Neighbor	4.48	16.90%	1 s	6.94	21.97%	3 s	9.68	24.74%	7 s
Pointer Network	3.83	0.00%	9 s	5.8	1.93%	2 min 4 s	8	3.09%	6 min 20 s

The outcomes presented in Table 2 unequivocally demonstrate the remarkable benefits of the Pointer Network architecture employed in this study during the comparative experiments involving the TSP. Particularly noteworthy is the Pointer Network's performance when pitted against professional solvers like Gurobi and OR-Tools. For TSP instances featuring 20 nodes, the Pointer Network effectively approximates the optimal solution cost and gap, all while maintaining a notably shorter runtime. As the problem scale expands to 50 and 100 nodes, the Pointer Network remains on par with Gurobi in terms of performance, even showcasing swifter execution times. In direct comparison with traditional heuristic algorithms, such as 2-opt and Nearest Neighbor, the Pointer Network consistently outshines across all problem scales. It consistently generates outcomes that closely approach optimal solutions while concurrently achieving higher time efficiency. In summation, the Pointer Network demonstrates remarkable efficacy in addressing small-scale TSP problems. This attribute renders it particularly suitable for real-time task scheduling and path planning, thereby augmenting the efficiency and precision of UAV mission services.

#### 4.6. System Testing

System testing is the final step after the completion of software development, where the entire software system is tested to verify its functionality, performance, reliability, security, and compliance with requirements. In system testing, testers follow a predefined test plan and test cases to simulate real user scenarios and comprehensively test the system to identify defects and vulnerabilities, ensuring the quality of the system. The test results

will be used to evaluate whether the system is ready for release and deployment in the production environment and will serve as a basis for future maintenance and upgrades.

This study will conduct functional testing on each of the system's functional modules to validate their alignment with the specified requirements. These encompass various modules, such as the situational awareness and scheduling module, task scheduling management module, and perception data management module. The chosen approach for functional testing in this study is the black-box testing method. Black-box testing effectively simulates the behavior of authentic users or external systems, thereby closely emulating the real-world usage scenarios of end-users. This methodology ensures a comprehensive assessment of the system's performance during actual operation, gauging its ability to reliably and stably respond to diverse inputs as intended. Additionally, black-box testing eliminates the necessity of delving deep into the intricate specifics of the system's internal implementation. This attribute enhances the simulation of a regular user's experiential interaction while proficiently capturing potential issues that might arise at the functional layer of the system. Another rationale underlying the adoption of black-box testing is its capacity to underscore the system's external behavior. The emphasis lies in assessing the system's response to distinct inputs and the corresponding output outcomes. This becomes pivotal for verifying the fulfillment of requirements within each functional module. A spectrum of scenarios, including boundary cases and exceptional circumstances, can be meticulously simulated, thus enabling an exhaustive evaluation of the system's performance and stability across diverse scenarios. This methodological approach significantly aids in the identification of latent functional defects, thereby facilitating the early resolution of potential concerns and consequently ensuring the system's reliability. In the process of designing test cases, functional requirements and specification documents serve as the guiding principles. By inputting varied data and operating the system, these test cases ascertain whether the system generates accurate outputs in accordance with expectations. Test cases of situation awareness and scheduling module are shown in Table 3. Test cases of task allocation and route planning module are shown in Table 4. Test cases of task scheduling management module are shown in Table 5. Test cases of data and device management module are shown in Table 6.

**Table 3.** Test cases of situation awareness and scheduling module.

Number	Test Steps	Expected Results	Test Results
1	Start the system and go to the dashboard page.	The dashboard displays real-time data of the UAV.	Consistent with expected results.
2	Initiate an image return request and observe the image results of the image return.	Receive images from UAV in real time.	Consistent with expected results.
3	View the UAV on the map display page location.	Real-time display of UAV positions on the map.	Consistent with expected results.
4	Send control commands.	The UAV executes on command.	Consistent with expected results.
5	Check that the warning message module is working properly.	Real-time display of UAV warning information.	Consistent with expected results.

**Table 4.** Test cases of task allocation and route planning module.

Number	Test Steps	Expected Results	Test Results
1	Provides inputs containing multiple UAVs and mission Input Parameters.	Return the correct task assignment result.	Consistent with expected results.
2	Provide an empty UAV as input parameter.	Return empty task assignment results.	Consistent with expected results.
3	Provide invalid UAV or tasks as an input parameter.	Returns an error message.	Consistent with expected results.
4	Provides input parameters with a large number of tasks.	Algorithm completes task assignment within reasonable time.	Consistent with expected results.
5	Test the performance and stability of algorithms.	Algorithms are able to process a large tasks in a reasonable amount of time and return correct results.	Consistent with expected results.

**Table 5.** Test cases of task scheduling management module.

Number	Test Steps	Expected Results	Test Results
1	Viewing Pending Tasks on the Task Assignment Screen list.	Displays all pending assignments in the system.	Consistent with expected results.
2	Click Task Assignment on the Task Assignment page.	Call the task assignment algorithm and obtain the result. display	Consistent with expected results.
3	Viewing Assigned Tasks in the Task Execution List tasks in the task execution list.	Displays a list of tasks that have been assigned to the UAV.	Consistent with expected results.
4	Check the mission visualization module to see the UAV mission execution routes and mission completion.	Displays the route of the UAV's mission and mission completion completion of the mission.	Consistent with expected results.
5	Add new tasks and validate task data addition and query functions.	After adding a new task, you are able to query the added tasks and verify the accuracy of task information.	Consistent with expected results.
6	Modify existing tasks and validate tasks Data modification and query functions.	After modifying an existing task, you will be able to query the modified task information and verify the accuracy of the information. accuracy.	Consistent with expected results.
7	Delete existing tasks and validate tasks Data deletion and query functions.	After deleting an existing task, it is not possible to query the Deleted task information.	Consistent with expected results.

**Table 6.** Test cases of data and device management module.

Number	Test Steps	Expected Results	Test Results
1	Ensure that the system is able to properly collect UAV data of all types and store it.	UAV data were successfully collected and stored.	Consistent with expected results.
2	Add a new data/device.	The data/device was successfully added and can be queried to the added data/device.	Consistent with expected results.
3	Query existing data/devices.	Returns correct data/device information.	Consistent with expected results.
4	Modify existing data/device information.	Data/device information is modified successfully and the modified data/device can be queried.	Consistent with expected results.
5	Delete existing data/devices.	Data/device deleted successfully.	Consistent with expected results.
6	Query all data/devices.	Returns all data/device information in the system information in the system.	Consistent with expected results.
7	Add duplicate data/equipment.	Failure to add data/device, the system gives a corresponding error The system gives a corresponding error message.	Consistent with expected results.

By conducting tests on each functional module, the output results of the main functions in each module match the expected results. This confirms that all functional modules are operating correctly, and the system has passed the testing phase.

## 5. Conclusions

This article explores the rapid development and widespread application of UAV technology, emphasizing the challenges faced by UAV clusters in terms of task allocation efficiency and data transmission security. To address these issues, this study combines blockchain technology with multi-drone situational awareness and successfully designs and implements a blockchain-based multi-UAV task processing system for situation awareness and real-time decisions, which has a positive impact on sustainable transportation. Through experiments, it has been proven that our system has achieved improvements in task efficiency. Through real-time task scheduling and collaborative processing, we have successfully improved the efficiency of task allocation, overcoming traditional bottlenecks and providing vital support for the efficient operation of sustainable transportation. This intelligent resource allocation helps reduce energy wastage and environmental impacts. Secondly, we have reinforced data security. By storing task information in blockchain smart contracts, we ensure the immutability and verifiability of task data, enhancing transparency and reliability in task allocation. This critical safeguard provides assurance for

data security and integrity in sustainable transportation, helping to prevent potential data breaches and risks. Additionally, our system introduces autonomous collaboration mechanisms. Through the automatic execution of smart contracts, the system autonomously coordinates task allocation and execution, enhancing processing efficiency and accuracy. This is of paramount importance for autonomous collaboration and resource optimization in sustainable transportation, contributing to improved transportation efficiency and reduced congestion.

However, despite these remarkable achievements, it is essential to acknowledge that there is still much work to be done. Future work includes further enhancing the stability and robustness of the system, especially in complex environments, to adapt to diverse environmental changes and emergency situations. Furthermore, continuous optimization of system performance is required to address performance bottlenecks and scalability limitations in handling large-scale tasks and multi-UAV collaborative operations. With the ongoing development in the field of sustainable transportation, we are committed to extending the outcomes of this research to broader application areas to meet the growing demands.

**Author Contributions:** Conceptualization, Z.C. and X.X.; methodology, Z.C. and X.X.; software, W.W. and Y.X.; validation, W.W. and Y.X.; formal analysis, Z.C.; investigation, W.W.; resources, O.A.; data curation, Z.C.; writing—original draft preparation, Z.C.; writing—review and editing, X.X. and Y.X.; visualization, W.W.; supervision, X.X. and O.A.; project administration, X.X.; funding acquisition, O.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the Researchers Supporting Project Number (RSP2023R102), King Saud University, Riyadh, Saudi Arabia.

**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. Ning, Z.; Dong, P.; Wang, X.; Hu, X.; Guo, L.; Hu, B.; Guo, Y.; Qiu, T.; Kwok, R. Mobile Edge Computing Enabled 5G Health Monitoring for Internet of Medical Things: A Decentralized Game Theoretic Approach. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 463–478. [\[CrossRef\]](#)
2. Wang, W.; Li, X.; Xie, L.; Lv, H.; Lv, Z. Unmanned Aircraft System Airspace Structure and Safety Measures Based on Spatial Digital Twins. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2809–2818. [\[CrossRef\]](#)
3. Ning, Z.; Yang, Y.; Wang, X.; Song, Q.; Guo, L.; Jamalipour, A. Multi-agent deep reinforcement learning based uav trajectory optimization for differentiated services. *IEEE Trans. Mob. Comput.* **2023**, 1–17. [\[CrossRef\]](#)
4. Sun, L.; Wan, L.; Wang, J.; Lin, L.; Gen, M. Joint Resource Scheduling for UAV-Enabled Mobile Edge Computing System in Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, 1–9. [\[CrossRef\]](#)
5. Zhu, B.; Bedeer, E.; Nguyen, H.H.; Barton, R.; Gao, Z. UAV Trajectory Planning for AoI-Minimal Data Collection in UAV-Aided IoT Networks by Transformer. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 1343–1358. [\[CrossRef\]](#)
6. Ning, Z.; Chen, H.; Ngai, E.C.H.; Wang, X.; Guo, L.; Liu, J. Lightweight Imitation Learning for Real-Time Cooperative Service Migration. *IEEE Trans. Mob. Comput.* **2023**, 1–18. [\[CrossRef\]](#)
7. Feng, C.; Liu, B.; Guo, Z.; Yu, K.; Qin, Z.; Choo, K.K.R. Blockchain-Based Cross-Domain Authentication for Intelligent 5G-Enabled Internet of UAVs. *IEEE Internet Things J.* **2022**, *9*, 6224–6238. [\[CrossRef\]](#)
8. Ning, Z.; Hu, H.; Wang, X.; Guo, L.; Guo, S.; Wang, G.; Gao, X. Mobile Edge Computing and Machine Learning in The Internet of Unmanned Aerial Vehicles: A Survey. *ACM Comput. Surv.* **2023**, *56*, 1–31. [\[CrossRef\]](#)
9. Meng, K.; He, X.; Wu, Q.; Li, D. Multi-UAV Collaborative Sensing and Communication: Joint Task Allocation and Power Optimization. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 4232–4246. [\[CrossRef\]](#)
10. Adil, M.; Jan, M.A.; Liu, Y.; Abulkasim, H.; Farouk, A.; Song, H. A Systematic Survey: Security Threats to UAV-Aided IoT Applications, Taxonomy, Current Challenges and Requirements With Future Research Directions. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 1437–1455. [\[CrossRef\]](#)
11. Ning, Z.; Zhang, K.; Wang, X.; Guo, L.; Hu, X.; Huang, J.; Hu, B.; Kwok, R.Y.K. Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2212–2225. [\[CrossRef\]](#)



12. Fan, Q.; Ansari, N. Towards Traffic Load Balancing in UAV-Assisted Communications for IoT. *IEEE Internet Things J.* **2019**, *6*, 3633–3640. [\[CrossRef\]](#)
13. Chen, C.; Zhang, Y.; Khosravi, M.R.; Pei, Q.; Wan, S. An Intelligent Platooning Algorithm for Sustainable Transportation Systems in Smart Cities. *IEEE Sens. J.* **2021**, *21*, 15437–15447. [\[CrossRef\]](#)
14. Wang, X.; Ning, Z.; Guo, S.; Wang, L. Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 598–611. [\[CrossRef\]](#)
15. Linthicum, D.S. The Technical Case for Mixing Cloud Computing and Manufacturing. *IEEE Cloud Comput.* **2016**, *3*, 12–15. [\[CrossRef\]](#)
16. Wang, X.; Li, J.; Ning, Z.; Song, Q.; Guo, L.; Guo, S.; Obaidat, M.S. Wireless Powered Mobile Edge Computing Networks: A Survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [\[CrossRef\]](#)
17. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 4804–4814. [\[CrossRef\]](#)
18. Aly, M.; Khomh, F.; Guéhéneuc, Y.G.; Washizaki, H.; Yacout, S. Is Fragmentation a Threat to the Success of the Internet of Things? *IEEE Internet Things J.* **2019**, *6*, 472–487. [\[CrossRef\]](#)
19. Ning, Z.; Xia, F.; Ullah, N.; Kong, X.; Hu, X. Vehicular Social Networks: Enabling Smart Mobility. *IEEE Commun. Mag.* **2017**, *55*, 16–55. [\[CrossRef\]](#)
20. Wan, L.; Li, X.; Xu, J.; Sun, L.; Wang, X.; Liu, K. Application of Graph Learning With Multivariate Relational Representation Matrix in Vehicular Social Networks. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 2789–2799. [\[CrossRef\]](#)
21. Ning, Z.; Dong, P.; Wang, X.; Rodrigues, J.J.P.C.; Xia, F. Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–24. [\[CrossRef\]](#)
22. Chen, J.; Zhang, Y.; Teng, S.; Chen, Y.; Zhang, H.; Wang, F.Y. ACP-Based Energy-Efficient Schemes for Sustainable Intelligent Transportation Systems. *IEEE Trans. Intell. Veh.* **2023**, *8*, 3224–3227. [\[CrossRef\]](#)
23. Xie, H.; Zheng, J.; He, T.; Wei, S.; Shan, C.; Hu, C. B-UAVM: A Blockchain-supported Secure Multi UAV Task Management Scheme. *IEEE Internet Things J.* **2023**, *1*. [\[CrossRef\]](#)
24. Wang, X.; Ning, Z.; Guo, S.; Wen, M.; Guo, L.; Poor, H.V. Dynamic UAV Deployment for Differentiated Services: A Multi-Agent Imitation Learning Based Approach. *IEEE Trans. Mob. Comput.* **2023**, *22*, 2131–2146. [\[CrossRef\]](#)
25. Khan, M.A.; Ullah, I.; Alkhalifah, A.; Rehman, S.U.; Shah, J.A.; Uddin, M.I.; Alsharif, M.H.; Algarni, F. A Provable and Privacy-Preserving Authentication Scheme for UAV-Enabled Intelligent Transportation Systems. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3416–3425. [\[CrossRef\]](#)
26. Mozaffari, M.; Saad, W.; Bennis, M.; Nam, Y.H.; Debbah, M. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2334–2360. [\[CrossRef\]](#)
27. Ning, Z.; Huang, J.; Wang, X. Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities. *IEEE Wirel. Commun.* **2019**, *26*, 87–93. [\[CrossRef\]](#)
28. Wang, X.; Ning, Z.; Wang, L. Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4568–4578. [\[CrossRef\]](#)
29. Ning, Z.; Sun, S.; Wang, X.; Guo, L.; Wang, G.; Gao, X.; Kwok, Y.K. Intelligent resource allocation in mobile blockchain for privacy and security transactions: A deep reinforcement learning based approach. *Sci. China Inf. Sci.* **2021**, *64*, 162303. [\[CrossRef\]](#)
30. Shaikh, A.; Goyal, M.; Greenberg, A.; Rajan, R.; Ramakrishnan, K. An OSPF topology server: Design and evaluation. *IEEE J. Sel. Areas Commun.* **2002**, *20*, 746–755. [\[CrossRef\]](#)
31. Laghate, M.; Cabric, D. Learning Wireless Networks' Topologies Using Asymmetric Granger Causality. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 233–247. [\[CrossRef\]](#)
32. Baek, H.; Lim, J. Design of Future UAV-Relay Tactical Data Link for Reliable UAV Control and Situational Awareness. *IEEE Commun. Mag.* **2018**, *56*, 144–150. [\[CrossRef\]](#)
33. Li, P.; Wang, L.; Wu, W.; Zhou, F.; Wang, B.; Wu, Q. Graph neural network-based scheduling for multi-UAV-enabled communications in D2D networks. *Digit. Commun. Netw.* **2022**. [\[CrossRef\]](#)
34. Tian, S.; Li, Y.; Zhang, X.; Zheng, L.; Cheng, L.; She, W.; Xie, W. Fast UAV path planning in urban environments based on three-step experience buffer sampling DDPG. *Digit. Commun. Netw.* **2023**. [\[CrossRef\]](#)
35. Li, M.; He, S.; Li, H. Minimizing Mission Completion Time of UAVs by Jointly Optimizing the Flight and Data Collection Trajectory in UAV-Enabled WSNs. *IEEE Internet Things J.* **2022**, *9*, 13498–13510. [\[CrossRef\]](#)
36. Liu, K.; Yan, Z.; Liang, X.; Kantola, R.; Hu, C. A survey on blockchain-enabled federated learning and its prospects with digital twin. *Digit. Commun. Netw.* **2022**. [\[CrossRef\]](#)
37. Ning, Z.; Sun, S.; Wang, X.; Guo, L.; Guo, S.; Hu, X.; Hu, B.; Kwok, R.Y.K. Blockchain-Enabled Intelligent Transportation Systems: A Distributed Crowdsensing Framework. *IEEE Trans. Mob. Comput.* **2022**, *21*, 4201–4217. [\[CrossRef\]](#)
38. Wang, X.; Zhu, H.; Ning, Z.; Guo, L.; Zhang, Y. Blockchain Intelligence for Internet of Vehicles: Challenges and Solutions. *IEEE Commun. Surv. Tutor.* **2023**. [\[CrossRef\]](#)
39. Alladi, T.; Chamola, V.; Sahu, N.; Guizani, M. Applications of blockchain in unmanned aerial vehicles: A review. *Veh. Commun.* **2020**, *23*, 100249. [\[CrossRef\]](#)
40. Miao, Q.; Lin, H.; Hu, J.; Wang, X. An intelligent and privacy-enhanced data sharing strategy for blockchain-empowered Internet of Things. *Digit. Commun. Netw.* **2022**, *8*, 636–643. [\[CrossRef\]](#)

41. Ma, S.; Wang, S.; Tsai, W.T. Delay Analysis of Consensus Communication for Blockchain-Based Applications Using Network Calculus. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 1825–1829. [[CrossRef](#)]
42. Seid, A.M.; Lu, J.; Abishu, H.N.; Ayall, T.A. Blockchain-Enabled Task Offloading With Energy Harvesting in Multi-UAV-Assisted IoT Networks: A Multi-Agent DRL Approach. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3517–3532. [[CrossRef](#)]
43. Campos, M.; Ponzoni Carvalho Chanel, C.; Chauffaut, C.; Lacan, J. Towards a Blockchain-Based Multi-UAV Surveillance System. *Front. Robot. AI* **2021**, *8*, 557692. [[CrossRef](#)]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.