

High Availability Migration Mechanism for UAV Failures in Edge Scenarios

Haoran Jin, Yi Li, Shengye Pang, Jinsong Shu, Chen Zhi, Xuhong Zhang, Yueshen Xu, Junlan Cai, Shaofeng Liu, Xiang Li, and Xinkui Zhao*

Abstract: Unmanned Aerial Vehicles (UAVs) are widely used in Internet of Things (IoT) scenarios, but factors like circuit malfunctions and network issues can lead to data loss, disconnections, and service interruptions, causing UAVs to go offline and fail to meet user needs. This problem can be effectively solved using UAV resource migration techniques. However, traditional methods lead to resource redundancy and cannot ensure effective migration decisions or the continuity of services. Therefore, we proposed a new UAV framework with a high-availability migration strategy to improve UAV swarm resource utilization. By combining Software Defined Network and Mobile Edge Computing technologies, we designed a UAV collaboration architecture model and introduced large language model agents in the edge layer for better task execution. Additionally, we created a two-tier collaborative UAV high-availability mechanism for handling failures in edge scenarios during UAV collaborative tasks. Our experiments on the container cloud platform showed a 42% reduction in failure recovery time and higher overall resource utilization compared to baseline algorithms. While our focus is on UAVs, our findings can potentially be applied to broader IoT and mobile edge computing scenarios such as traffic management systems and healthcare IoT systems.

Key words: edge computing; Unmanned Aerial Vehicle (UAV) collaboration; task migration; high availability

1 Introduction

The proliferation of 5th Generation (5G) mobile communication technology creates a fast, stable, low latency, high bandwidth, and massive connectivity

network connection for the development of Internet of Things (IoT) and supports the simultaneous connection of a large number of devices, which enables decentralized, real-time processing at the network edge, leading to faster, more efficient, and more secure

- Haoran Jin is with the Zhejiang University-University of Illinois Urbana-Champaign Institute, Zhejiang University, Jiaxing 314499, China. E-mail: haoranj.21@intl.zju.edu.cn.
- Yi Li, Jinsong Shu, Chen Zhi, Xuhong Zhang, and Xinkui Zhao are with School of Software Technology, Zhejiang University, Ningbo 315048, China. E-mail: 22251190@zju.edu.cn; 22351172@zju.edu.cn; zjuzhichen@zju.edu.cn; zhangxuhong@zju.edu.cn; zhaoxinkui@zju.edu.cn.
- Shengye Pang is with College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China. E-mail: pangsy@zju.edu.cn.
- Yueshen Xu is with School of Computer Science and Technology, Xidian University, Xi'an 710126, China. E-mail: ysxu@xidian.edu.cn.
- Junlan Cai, Shaofeng Liu, and Xiang Li are with the Beijing Life Science Academy, Beijing 102200, China, and also with Zhengzhou Tobacco Research Institute of China National Tobacco Corporation, Zhengzhou 450001, China. E-mail: caijunlan@sina.com; sfliu@126.com; lixiang79ben@sina.com.

* To whom correspondence should be addressed.

Manuscript received: 2024-05-31; revised: 2024-07-11; accepted: 2024-07-20

IoT applications. IoT connects and exchanges data through communication networks such as the Internet and can access a large amount of information and data from a large number of edge devices. According to Gartner, by 2025, 75% of data is expected to be created at the edge side of the IoT^[1], outside of centralized data centers. However, the explosion of data will result in much longer transmission times, and even with high-bandwidth networks, it will still take a long time to transmit such massive amounts of data. Whereas most of the IoT application services require low-latency network response, leading to the fact that the traditional computing method of transmitting all the data to the cloud can no longer meet the demand. Therefore, in order to reduce the pressure on network bandwidth and the latency of data transmission, the computation and processing tasks need to be decentralized to edge nodes that are closer to the location where the data is generated, which has given rise to edge computing. Edge computing has a wide range of applications in scenarios such as urban intelligence, industrial manufacturing intelligence^[2], and smart buildings. In these scenarios, multi-device collaboration is becoming more and more common due to its ability to improve the performance, reliability, security, and efficiency of the system. The common collaboration methods are categorized into collaboration between cloud and edge devices and collaboration between edge devices. In cloud-edge device collaboration, edge devices first perform initial data processing and selection, then transmit the streamlined data to the cloud. This process effectively reduces the computational load on the cloud, allowing for more efficient data management and faster processing speeds. On the other hand, collaboration among edge devices involves multiple edge nodes or devices working in concert to accomplish specific tasks. This often includes the sharing of data, resources, and computational duties, enabling more resilient and adaptive operations. By distributing tasks across several nodes, this collaborative approach not only minimizes the risk of system failure due to any single point of failure but also enhances overall task execution speed and reliability.

In IoT scenarios, Unmanned Aerial Vehicle (UAV) is a common edge device that can accomplish tasks that humans cannot by carrying multiple sensors such as infrared camera and performing specific tasks^[3].

Common scenarios include monitoring traffic flow, road conditions, disaster monitoring and response. These scenarios are common in smart factories and smart cities, where UAVs can play a big role. However, a single UAV often cannot achieve full coverage, and thus multiple UAVs are required to work together as edge devices. This technique, also known as multi-UAV collaboration, is a common and efficient technique in IoT environments, with applications including target tracking, surveillance, and remote sensing. With UAVs working together, we can improve the performance, reliability and efficiency of UAV applications, especially in scenarios that require instantaneous decision making and low-latency response, such as mission monitoring, search and rescue, and so on. At the same time, the combination of UAVs and edge computing can also reduce the cost of data transmission and processing^[4], improve data privacy and security^[5]. The existing working architecture of the UAV swarm mainly consists of two parts, the edge and the cloud^[6]. The edge end consists of UAVs and edge server nodes. The UAV acts as an edge device carries sensors to accomplish tasks, and the edge server node has computing and memory capabilities, which can provide service resources for the devices within its signal coverage. The cloud is responsible for processing the information from the edge, and selecting the appropriate UAV to send the task as well as acting as a strong support when the computing load of the edge is insufficient.

However, in daily operation, the whole architecture needs to deal with a variety of unexpected situations, which leads to the failure or offline of UAVs and edge server nodes very common, affecting the efficiency of task completion, which brings a lot of trouble to the users. Common problems include failures due to physical conditions, failures due to software, server overload and so on^[7]. Physical failure may cause as a result of malfunctioning communications equipment, power problems or beyond communication boundaries^[8]. These are particularly common in remote areas and rural areas due to the lack of communication infrastructure, since UAVs may face frequent disconnections due to weak or non-existent network signals, leading to increased downtime and reduced operational effectiveness. For the software aspect, there may be problems with the communication software, operating system or communication protocols of the

UAV, such as communication timeouts, excessive loads, and network delays, etc.^[9] Software bugs or glitches can cause UAVs to lose connectivity with the control platform or fail to execute tasks as programmed. These factors often result in one or more UAVs being disconnected from the entire existing UAV control platform. Currently, resource scheduling and migration have been chosen to solve this problem^[10,11].

However, existing UAV scheduling strategies suffer from significant problems, since they can not dynamically adjusting the number of UAVs allocated to missions based on real-time data analysis, which cause a large number of redundant resources during deployment, resulting in high costs and low resource utilization rate^[12]. This produces a significant waste of equipment and energy resources, which needs to be urgently addressed as UAV technology has a wide range of application scenarios in numerous domains and people often cannot afford the cost associated with so many redundant resources. Therefore, how to generate high-availability and efficient resource scheduling strategy to provide users with stable services in edge scenarios has become an important issue to improve UAV service quality and user satisfaction.

In order to solve those problems brought to users by the failure of UAV architecture with high efficiency, our work optimized the existing resource scheduling methods, and tackled the problems that could not be solved by the existing migration strategies, such as the difficulty of achieving a comprehensive measurement of dynamic parameters such as the state of the UAVs and the lack of assessment of the future service capability of the edge nodes, and then reduced the redundancy of resources and achieves a high resource utilization rate. After that, we proposed distributed scheduling algorithms and centralized scheduling algorithms for different scenarios by targeting the life cycle of the tasks as an indicator, and then selected the most suitable migration objects to achieve more optimization of the baseline scheduling^[13].

The primary contributions of this paper are as follows:

- We improved the existing “UAV+Cloud” architecture into a bottom-up smart UAV collaboration architecture covering the edge, control and cloud layers. The control layer consists of multiple UAV

controllers, each of which corresponds to a region constituted by an edge device, so that we can realize direct communication between UAVs in the same region, and cross-region communication of UAVs is realized through the control layer as a hub.

- We proposed a strategy for a double-layer high availability mechanism that includes distributed scheduling algorithms and centralized scheduling algorithms for different scenarios for the failure of UAVs and edge server nodes in edge scenarios based on the architecture, where the control layer decides whether to realize same-region or cross-region task succession based on the states of the UAVs and nodes.

- Analytical and comparative experiments were conducted to demonstrate the effectiveness of the proposed migration strategy.

The rest of the paper is organized as follows. Section 2 describes the related work of this paper, and Section 3 introduces the system model and formulates the problem. Section 4 presents our proposed methodology. Section 5 discusses the simulation results. Finally, the paper is concluded in Section 6.

2 Related Work

The related works for our paper is divided into two main aspects of content, the first aspect is the application of collaborative work of UAVs in IoT edge scenarios, and the second aspect is the existing UAV migration and scheduling technology in IoT edge scenarios.

With the continuous progress of IoT technology and the increasing maturity of UAV technology, the UAV collaboration has a wide range of application prospects. Yazid et al.^[14] focused on issues such as computational offloading and latency in mobile edge computing scenarios utilizing multi-UAVs collaboration and described the advantages and future of multi-UAVs as device collaboration in mobile edge computing. Motlagh et al.^[14] presented a high-level view of an integrated drone-based IoT platform for delivering IoT services from high altitudes and the coordination of the entire system, demonstrating how to utilize UAV collaboration for face-recognition-based crowd-surveillance, which was found to be very effective in shortening the recognition processing time and detecting suspicious persons in a timely manner. Alotaibi et al.^[15] used the autonomous UAVs coordinated with each other to accomplish the search

and rescue mission in the shortest possible time while saving as many survivors as possible, which significantly increased efficiency compared to the traditional UAV search and rescue model. Qi et al.^[16] had conducted research on the concept of an Internet in the air, hoping to realize global unification through three-dimensional connected heterogeneous smart devices. In their design, UAVs are connected to multiple base stations on the ground to realize the information transfer between UAVs and nodes, thus forming a 5G layered IoT network in the air. Dong et al.^[17] categorized the UAVs in a UAV swarm into two types of UAVs based on their functions as user nodes and as edge servers, described the roles of each functional UAV in the swarm, and discussed how to reduce the data transmission latency and improve the efficiency of the UAVs through a collaborative approach. McEnroe et al.^[18] expanded on this classification and discussion by discussing the impact of artificial intelligence on collaborative work under edge computing and intelligent combination, and analyzed the impact of the key technique of UAVs, such as autonomous navigation.

Therefore, we need UAV migration scheduling to realize the migration process and complete the migration more efficiently, in order to optimize the scheduling resources, satisfying user needs and maintaining quality of service, academics have studied the following two approaches. The first approach is exact and approximate solution algorithm for the optimal resource scheduling problem. Exact solution methods are algorithms in which the optimal analytical solution of the problem can be obtained through mathematical modeling. These methods typically transform the resource scheduling problem into an optimization process under a Mixed Integer Programming (MIP) model, which is then solved using a branch-and-bound algorithm^[19]. However, the complexity of branch-and-bound algorithms grows exponentially with the problem size and cannot solve large-scale problems in an acceptable time frame. Approximate solution methods are methods that approximate the solution of resource scheduling optimization problems in polynomial time. Common solution algorithms include branch-and-bound, cut-plane method^[20], relaxation algorithms, and dynamic programming^[21]. However, when the size of the UAV resource scheduling optimization problem is large, the

recovery time of these methods becomes long. In addition, not all complex resource optimization scheduling problems can be represented by linear or convex constraints, which makes it difficult to model and solve these problems using linear programming and convex optimization methods. The second approach is heuristic solution algorithms for resource optimization scheduling problems, which aim at solving the problem through intuitive or empirical methods in order to obtain a relatively optimal solution within an acceptable computational cost. However, it is difficult to predict the degree of approximation between the solution obtained by the algorithm and the optimal solution during the solution process. Common heuristic algorithms include Taboo Search (TS)^[22], Simulated Annealing (SA)^[23], Genetic Algorithm (GA)^[24], Ant Colony Optimization (ACO)^[25], Particle Swarm Optimization (PSO)^[26], and Artificial Neural Networks (ANN)^[27]. For example, the problem of task sequencing and scheduling for n jobs and m machines in a batch processing environment was solved in Ref. [28] using an ACO algorithm. A hybrid algorithm combining genetic algorithm and simulated annealing was proposed in Ref. [29] to study the two-phase assembly hybrid shop scheduling problem with job shift constraints. The author improved the “short-sighted view” problem in the heuristic algorithms that can further improve the accuracy of the algorithms^[30] proposed the principle of energy-neutral flight and distribution scheduling algorithm, which enables UAVs to plan flight paths without exhausting energy, and solves the problem of insufficient flight range and load capacity. In order to ensure service continuity, Ref. [31] proposed a distributed task migration optimization algorithm based on deep reinforcement learning, the authors successfully constructed an Adaptive Weighted Deep Deterministic Policy Gradient (AWDDPG) algorithm to optimize the migration cost and service latency. Reference [32] designed a two-tier distributed Software Defined Network (SDN) control network architecture, which uses federated blockchain, described the model of the service offloading and migration optimization problem as an MDP model, and designed a dynamic service offloading and migration algorithm based on a deep reinforcement learning algorithm to solve it. Aiming at the adaptive service migration process in device networks, a deep learning service migration decision-

making algorithm is proposed in Refs. [33,34], and the algorithm can dynamically adjust the device service migration decision according to the environmental information.

Although these works have achieved considerable results, there are still some problems. First, when studying migration after cooperative UAV localization, most UAV migration scenarios are difficult to achieve a comprehensive measurement of dynamic parameters such as UAV state, which cannot guarantee the effectiveness of migration decisions and the continuity of mobile services. In addition, most of the existing designs use proximity scheduling or simply make decisions based on the existing resource provisioning of edge servers, and the scheduling process is lack of evaluating the future service capability of edge nodes and the consideration of the forward scheduling process of service resources. Meanwhile, researchers tend to focus too much on the execution efficiency of a specific UAV collaborative task (or multiple UAVs) to establish a local optimization problem, while ignoring the impact of scheduling decisions on metrics such as resource utilization. Through redesigning the UAV working architecture and proposing an algorithm, our proposed solution enables comprehensive measurement of dynamic UAV parameters and future edge node service capability for high resource utilization.

3 System Model

In this section, we first introduce and explain the background knowledge of our collaborative architecture, such as how the migration process work and how we utilize the Mobile Edge Computing (MEC) and SDN technology. Then we introduce the migration and scheduling process in this paper by means of a motivating example. Finally, we establish the problem based on this motivating example.

3.1 Background knowledge

Device migration: Device migration in IoT refers to the process of transferring the operational state, data, and tasks of an IoT device from one physical location or network node to another. This process ensures continuity of service, efficient resource management, and adaptability to changing network conditions or device statuses. In this paper, we have two cases about migration. The first one refers to migrate the application from the failed UAV to the new UAV, and let the new one take over the functions. The second

refers to giving command or control ability of UAVs to another edge node when one edge node fails.

Software defined network: SDN is a network architecture that separates the control plane and data plane of the network. It utilizes a centralized controller to manage and configure network devices. MEC is a technology that pushes computational and storage resources to the edge of the network to reduce data transfer latency and improve service quality. The combination of SDN and MEC enables real-time optimization of Quality of Service (QoS), ensuring quick response and optimized processing of applications at the edge nodes. Besides, it can enable intelligent processing at the edge, which includes real-time analytics, decision-making, and optimization.

3.2 Motivating example

As shown in Fig. 1, we could to control multiple UAVs by means of nodes for each region, thus realized that the user can manage all the UAVs in the cloud, and when the UAVs go offline, the cloud realized the migration of resources between different UAVs by sending the resources of the failed UAVs to the selected UAVs to re-manage the data of the original UAVs as well as to take over the tasks of the original UAVs. This process includes resource preparation, data backup as shown in the bottom of Fig. 1. Migration could be triggered by a variety of reasons, such as system maintenance, resource optimization, and so on. Before device migration, we need to evaluate the goals and effects of the migration. Next, we should back up data to ensure data integrity and recovery ability. The data backup process includes creating a comprehensive backup of all critical data and system configurations. After that, we need to ensure there is a stable network connection between the two UAVs and verify the storage space on the target UAV. Next, we should use a secure file transfer protocol to transfer the data from the source UAV to the target UAV. Additionally, it is important to establish a recovery plan to restore data promptly in case of any issues during the migration process. Ensuring data integrity and recovery capability are crucial for minimizing downtime and maintaining operational continuity. After the device migration, we should also connect and test to ensure that the device operates normally and communicates with the network properly, and continuously monitor the status of the device to ensure that it could operate normally after the migration. By means of intelligent scheduling, we

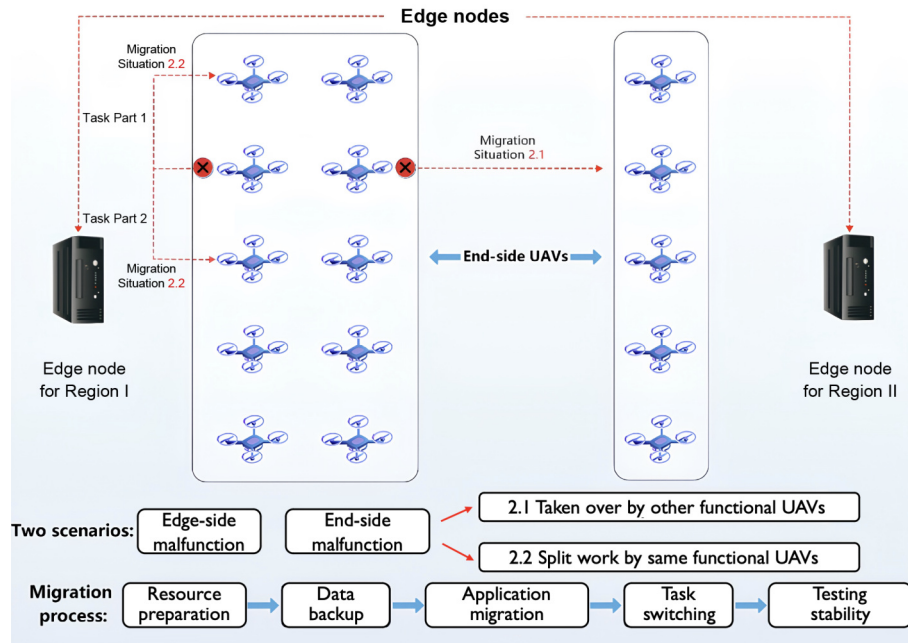


Fig. 1 Motivating example of the migration process in the edge layer.

could realize more efficient migrations, as shown by the red lines in Fig. 1, we could choose to migrate in the same region or between different regions. Intelligent scheduling refers to the rational allocation and scheduling of IoT devices, resources or tasks through the use of intelligent algorithms. These algorithms aim to optimize scheduling decisions by using real-time data, device status and task requirements to improve efficiency, reduce costs and satisfy various user needs. Common algorithms include shortest path algorithms, genetic algorithms, simulated annealing algorithms, ant colony algorithms, particle swarm algorithms and so on. These algorithms can be applied in different domains such as logistics, traffic management, manufacturing, city management, and disaster relief scenarios to improve network resource utilization, optimize routes and enhance system

performance, and improve task completion efficiency. The continuous development of intelligent scheduling of IoT devices will have a positive impact on the efficiency and sustainability of various domains. In the following part of the paper, we will introduce the proposed distributed scheduling algorithms and centralized scheduling algorithms. In the algorithm, by comparing the various migration parameters of different UAVs, the most suitable migration object is selected by comprehensive consideration to achieve a more optimized scheduling.

3.3 Problem establishment

As shown in Fig. 2, there are various scenarios and complex management and resource scheduling functions relied on by intelligence in the process of edge UAV collaboration. Based on the MEC and SDN

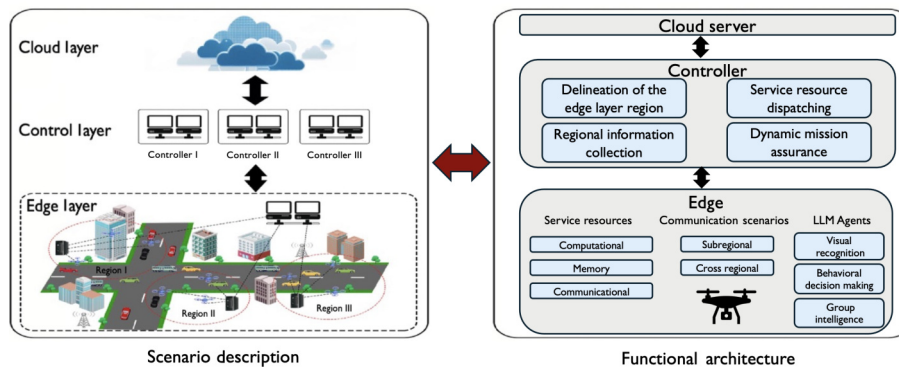


Fig. 2 Multi-UAV collaborative architecture for edge scenarios.

technology, we improved the existing “UAV+Cloud” architecture into an intelligent UAV collaboration architecture that covers the edge layer, the control layer and the cloud layer from the bottom up. The edge layer is mainly oriented to the basic network connection function of the UAV network, and the control layer supported by SDN technology is responsible for the information collection, data management, status monitoring and other functions in the network, and this paper adds a basic function module including UAV management and scheduling. The cloud layer, as an important support for the edge layer and the control layer when their capacity is insufficient, has powerful functions and provides rich resources and powerful services for the network connection and intelligence of the whole UAV network. Moving right along, we will detail the specific applications of each layer.

The edge layer is the main component of the intelligent UAV collaborative environment and contains various nodes such as UAVs and edge servers. The edge server provides service resources for users within its signal coverage area, and the UAVs realize the networking process such as information interaction and task collaboration in the edge layer through various communication methods such as wired or wireless. At the same time, we introduced Large Language Model (LLM) agents in the edge layer. We achieved it by selecting of a suitable LLM model, optimizing the model, selecting the operating system and drivers, integrating of the software, and programming of the application. By integrating with traditional computing elements in UAV networks (such as sensors, communication modules, edge computing devices, and cloud computing platforms) through data processing, natural language processing, autonomous decision support, fault detection and management and so on. For example, sensors such as cameras could provide real-time video and image data to the visual recognition LLM agent. This LLM could analyze environmental data, identifies targets and obstacles, and plans the best flight path by receiving the data, and generate commands that will be sent to the controller. Therefore, LLM agents could enhance the intelligence level of UAVs, improve their autonomy and collaboration capabilities, and enable them to better accomplish various complex tasks^[35]. For example, it could realize the visual recognition and analysis of the surrounding environment of the MEC through the visual recognition

intelligence, the behavioral decision-making and control of the MEC through the behavioral decision-making intelligence, and the group synergy and joint action of multiple MECs through the group intelligence, to guarantee that the final action results of the entire MEC cluster meet the control expectation. The algorithm used in the current large language model module is mainly based on the existing large language model to do scene adaptation and optimization, to provide basic support to meet the control objectives, this paper does not redevelop or train a new type of model for the demanded scene, so this paper does not do further in-depth analysis for the large language model module. In addition, in order to facilitate the management of the upper layer controller, the edge layer is divided into thousands of regions consisting of nodes such as UAVs and edge servers that are geographically close or categorized by tasks. Within the same region, nodes such as UAVs can communicate directly, and cross-region communication can be realized between different regions with the help of the control layer as a hub.

The control layer contains multiple controllers that are deployed in zones, and each controller is responsible for managing an edge layer region. In each region, the controller regularly collects and updates dynamic data such as resource status, UAV movement, etc., within the region through the communication process with device nodes such as UAVs. Based on the above information, the controllers are centrally responsible for basic intelligent functions such as edge UAV resource scheduling and task management within the region. The cloud layer is located in the top layer of the architecture. Existing cloud computing model related research and applications have been very mature, so this paper does not discuss the the specific functions of the cloud layer in depth, and mainly focuses on the UAV resource scheduling and dynamic service guarantee process in the lower and middle layers of the architecture.

In edge scenarios, there are two main situations encountered during UAV collaboration: UAV failure and edge node failure.

3.3.1 UAV failure

For the case of UAV failure, the recovery process mainly includes the control layer analyzing the global state for scheduling and UAV mission takeover. We call the failed UAV as UAV_broken, when there is a

UAV_broken exists, the control layer will first analyze the state of all UAVs in the same region, and selects a suitable UAV to take over the task; if there is no suitable UAV in the same region to replace it, then it reports to the cloud to analyze the entire UAV network, so as to select a suitable UAV in other regions to take over the task. The UAV task replacement includes the movement of the UAV to the designated location, the recovery of the task data between the UAV and the edge server. In addition to this, the succession of UAV missions between different regions also includes stages such as connecting to edge servers.

We calculated the UAV failure recovery time in several steps. Firstly, we considered UAVs moved to designated locations, when a UAV performs mission succession, the UAV_replace that takes over the mission needs to move to the specified location in order to perform operations such as connecting with the edge server and recovering mission data more smoothly. Assuming that the average speed of all UAVs is the same, the time delay_move for the replacement UAV to move to the specified location is positively correlated with the distance between the two UAVs, $s[\text{UAV_broken}, \text{UAV_replace}]$, which can be expressed as

$$\text{delay_move} = \alpha \cdot s[\text{UAV_broken}, \text{UAV_replace}] \quad (1)$$

where α is a constant that convert the distance s into a time delay. This proportionality constant takes into account all the factors that may affect the movement time, such as wind speed, obstacles in the path planning, etc.

Then, we considered the time it takes for the UAV to connect to the edge server is T_{conn} . When the UAV task succession involves scheduling between different regions, the UAV_replace that takes over the task needs to reconnect with the edge server within the specified region in order to successfully succeed the faulty UAV_broken task, and the time it takes for the UAV to reconnect to the edge server can be expressed as

$$\text{delay_conn} = \rho \cdot T_{\text{conn}} \quad (2)$$

$$\begin{cases} \rho = 0, & \text{when scheduling UAVs in the same region;} \\ \rho = 1, & \text{when scheduling UAVs between regions} \end{cases} \quad (3)$$

where ρ is the migration index, with 0 and 1 representing same-region migration and cross-region migration, respectively.

Finally, we consider the time of UAVs mission data recovery. After the UAV takes over the task, it needs to communicate with the edge server for data recovery and the time spent can be denoted as delay_recovery. To summarize, the sum of time for UAV failure recovery delay_uav is the sum of time spent on UAV movement, connection, and data recovery, which can be expressed as

$$\text{delay_uav} = \text{delay_move} + \text{delay_conn} + \text{delay_recovery} \quad (4)$$

3.3.2 Edge node failure

For the case of edge node failure, the UAV service assurance process also contains stages such as the control layer analyzing the global state for scheduling and UAV task succession. We called the broken edge node as node_broken. When it fails, the control layer will first analyze the state of the region, such as the node resource utilization rate, and select suitable edge nodes in the same region to replace; if there is no suitable edge node in the region to be replaced, then it will be reported to the cloud to analyze the whole node network, so as to select suitable edge nodes in other regions. The same as UAV failure, the service assurance process of edge node failure includes the complete phase of UAV mission takeover; the difference is that the failure recovery time of the edge node is the sum of the recovery time of the UAV swarm connection, which can be expressed as

$$\text{delay_node} = \text{delay_uav}_i + \dots + \text{delay_uav}_n \quad (5)$$

where i and n denote random numbers that represent the codes of the malfunctioning UAVs.

In our experiments, another experimental metric other than failure recovery time is overall resource utilization in the whole architecture. In our experiment, we will mainly consider service resources such as computing and memory resources. Firstly, we calculated the specific resource utilization rate. For a specific node a resource, the node that resource utilization rate for this node is the ratio of the actual value of the node's resources used at this time to the total value of the node's resources, which can be expressed as

$$\eta = \frac{\text{Used resources}}{\text{Total resources}} \quad (6)$$

Based on the specific resource utilization rate, we calculated the absolute error in resource utilization rate between nodes. For a cluster of nodes, too high or too

low utilization of a particular node is not conducive to the stability of the cluster. The absolute error of resource utilization rate between nodes is obtained by calculating the average value of the same resource utilization rate of different nodes, and then the absolute value of the error between the resource utilization rate of each node and the average value is sought in turn, in order to avoid wrong optimization caused by too large a gap in the number of nodes, we divided the sum of the absolute value of the error by the number of nodes, which can be expressed as follows:

$$\bar{\eta} = \frac{\eta_1 + \eta_2 + \dots + \eta_n}{n} \quad (7)$$

$$y = \frac{|\eta_1 - \bar{\eta}| + |\eta_2 - \bar{\eta}| + \dots + |\eta_n - \bar{\eta}|}{n} \quad (8)$$

In the end, we got the overall resource utilization rate in whole architecture. For the system, the overall resource utilization does not only involve a certain resource, but should cover a variety of resources, and different resources care about different degrees, so we assigned different weights β_i to different resources. The more the system cares, the higher the weight. So the overall resource utilization balance can be expressed as

$$y_{\text{sum}} = \beta_1 \cdot y_1 + \beta_2 \cdot y_2 + \dots + \beta_n \cdot y_n \quad (9)$$

where $y_1, y_2, \dots, y_n$ refer to the absolute errors in resource utilization of different resources of the node cluster, respectively. In a holistic view, the smaller the overall resource utilization balance is, the more stable the network is.

4 UAV Scheduling Strategy Based on Intelligent Algorithmic Approach

In this section, we establish an optimization function with the objective of minimizing the service failure recovery time while balancing the overall resource utilization, and under its guidance, we designed a two-tier collaborative UAV high-availability scheme containing two sub-algorithms (distributed scheduling sub-algorithm and centralized scheduling sub-algorithm) for the two failure scenarios.

4.1 High available strategy for UAV failures

First, we will introduce the distributed scheduling algorithm under this circumstance. It is initiated when the faulty UAV is under the same edge node ($\rho = 0$), i.e., a neighbor in the same region that has sufficient resources and satisfies the soft constraints. The

Algorithm 1 High availability algorithm for UAV failures (for a given region)

Input: UAV states, edge node states, resource proportion, and Working Soft Constraints (WSC) for failed UAVs;

Ensure: Create a scheduling priority queue Q , sorting by less recovery time and overall resource balance rate;

```

while Detecting UAV_broken do
  for UAVi under same edge node edgenodei do
    if UAV_state == on && UAVi satisfies WSC of UAV_broken then
      Add to candidate UAV collection S_candidate;
    end if
  end for
  if S_candidate! = none then                                ▶ Execute distributed
    scheduling algorithm
    for  $i = 1; i \leq S\_candidate.size; i++$  do
      Calculate t_pred from UAV_candidatei &
      UAV_broken state;
      Calculate resource utilization rate after
      UAV_candidatei migrated;
      Stored in dispatch priority queue  $Q$  for sorting;
    end for
    Select the target UAV_target at queue head;
    Move UAV_target to UAV_broken location; ▶ Initiate
    succession process;
    Completion of the transmission of tasks;
  else                                                        ▶ Execute centralized scheduling algorithm;
    Edgenodei that UAV_broken belongs reports failure to
    controller_a;
    for Each UAVi under controller_a do
      if UAV_state == on && UAVi satisfies WSC of
      UAV_broken then
        Add to candidate UAV collection S_candidate;
      end if
    end for
    if S_candidate! = none then
      Repeat second for loop;
    else
      No execution of UAV scheduling;
    end if
  end if
end while

```

implementation steps are detailed as Algorithm 1.

After a UAV failure is detected, the edge node constructs a distributed set of candidate UAVs, S_candidate, by comparing the amount of resources required for the failed UAV to perform its task and available resources amount of each UAV's neighbors using the information collected in its region of jurisdiction. Then, the edge node will pre-calculate the

objective function for each candidate UAV in the candidate set, budgets the service failure recovery time by comparing the relative geographic locations of the failed UAVs and the candidate UAVs, and calculates the overall resource utilization rate after migration by taking into account the overall resource utilization rate of the region, so as to locate the target UAVs. After that, a distributed UAV scheduling decision is initiated and the target UAV relocates to the original failed UAV for mission succession.

Then, we introduce the centralized scheduling algorithm. When the failed UAV is under different edge nodes, i.e., the set of candidate UAVs in the same region is empty, the centralized UAV scheduling sub-algorithm is initiated. The implementation steps are detailed as follows. After detecting a UAV failure and the candidate set is empty, the edge node will report the UAV failure information to the controller, which uses the information it has collected about the UAVs in the large area under its jurisdiction, compares the amount of resources required for the failed UAV to perform the task and the amount of resources available for all the UAVs in the area, and constructs the centralized candidate UAV set $S_{\text{candidate}}$. Then, the controller pre-calculates the objective function for each candidate UAV in the set, estimates the service failure recovery time by comparing the relative locations of the failed UAV and the candidate UAV, and calculates the overall resource utilization rate after migration by comprehensively considering the global resource utilization of the region, so as to locate the target UAV. After that, a centralized UAV scheduling decision is initiated, and the target UAV migrates to the original failed UAV to connect with the edge node and perform the mission takeover operation.

4.2 High available strategy for edge node failures

First, we introduce the distributed scheduling algorithm under this circumstance. It is initiated when the failed node is under the same controller ($\rho = 0$), i.e., a neighbor in the same large region that has sufficient resources and satisfies the soft constraints. The implementation steps are detailed as Algorithm 2.

The implementation steps are detailed as follows. After a node failure is detected, the controller constructs a distributed set of candidate nodes, $S_{\text{candidate}}$, by using the information it has collected about the nodes in its area of jurisdiction and comparing the amount of resources required by the

Algorithm 2 High availability algorithm for node failures (for a given region)

Input: UAV states, edge node states, resource proportion, and WSC for failed nodes;

Ensure: Create a scheduling priority queue Q , sorting by less recovery time and overall resource utilization rate;

```

while Detecting an edgenode_broken do
  for edgenodei under same controller_a do
    if edgenode_state == on && satisfies WSC of edgenode_broken then
      Add to candidate edge node collection  $S_{\text{candidate}}$ ;
    end if
  end for
  if  $S_{\text{candidate}} \neq \text{none}$  then ▷ Execute distributed scheduling algorithm;
    for  $i = 1; i \leq S_{\text{candidate}}.\text{size}; i++$  do
      Calculate  $t_{\text{pred\_sum}}$  for all UAV* migrate;
      Calculate overall utilization rate after all UAV* migrate;
      Stored in dispatch priority queue  $Q$  for sorting
    end for
    Select the scheduling target edgenode_target at queue head;
    Move all UAV* to edgenode_target location; ▷ Initiate succession process;
    UAV swarms linking target nodes;
    Completion of the transmission of tasks;
  else ▷ Execute centralized scheduling algorithm
    Controller that edgenode_broken belongs reports failure to cloud;
    for Each edge node under all controller do
      if edgenode_state == on && satisfies WSC of edgenode_broken then
        Add to candidate edgenode collection  $S_{\text{candidate}}$ ;
      end if
    end for
    if  $S_{\text{candidate}} \neq \text{none}$  then
      Repeat second for loop;
    else
      No execution of UAV scheduling;
    end if
  end if
end while

```

node to control the swarm of UAVs with the amount of resources available to each of the node's neighbors. Besides, we denoted UAVs under edgenode_broken as UAV*. Then, the controller will pre-calculate the objective function for each candidate node in the candidate set, estimates the maximum value of the service failure recovery time by comparing the state of the failed node with the candidate node and controlling

the UAV swarms, and calculates the overall resource utilization rate after migration by considering the regional resource utilization rate in a comprehensive manner, so as to locate the target node. After that, the distributed node scheduling algorithm will be initiated to migrate the swarm of UAVs under the control of the failed node to the target node to reconnect and take over the mission.

Then, we introduce the centralized scheduling algorithm. When the failed node is under the same controller, i.e., the set of candidate nodes in the same region is empty, the centralized node scheduling sub-algorithm is initiated. The implementation steps are detailed as follows. After detecting a node failure and the candidate set is empty, the controller sends a failure message to the cloud, which constructs a centralized candidate node set $S_candidate$ by analyzing the collected overall node information and comparing the amount of resources required by the failed node to control the UAV swarm with the amount of resources available for all nodes. Then, the cloud will pre-calculate the objective function for each candidate node in the candidate set, estimates the service failure full recovery time by comparing the state of the failed node with that of the candidate node, and calculates the overall resource utilization rate after the migration by taking into account the region resource utilization rate, so as to locate the target node. After that, the centralized node scheduling algorithm will be initiated to migrate the UAV swarm under the control of the failed node to the target node, reconnect, and take over the mission.

5 Experiment

Simulation experiments were conducted on the proposed UAV high availability solution on the container cloud platform, which is a container technology-based cloud computing platform designed to simplify and optimize the deployment, management and scaling of containerized applications. We detail the performance comparison and analysis, which are given on two metrics, namely, average service failure recovery time and overall resource utilization rate, as well as the simulated scheduling solution of the system in the case of failure.

5.1 Experiment setup

In the experimental scenario, we randomly distributed tasks to UAVs. Completion on the UAVs can be done

Table 1 Experimental parameters.

Parameter	Value
S (m ²)	120 000–840 000
N_c	2–7
N_n	5–100
N_{uav}	10–700
v (km/h)	70
M (GB)	0–150

either at the computational task or at the node. The UAV was assumed to fly at a constant altitude of 150 m. Table 1 contains a summary of the flying, computation, and communication parameters, where S represents experimental simulation scope; N_c , N_n , and N_{uav} represents number of controllers, nodes, UAVs, respectively; v represents the speed of UAV; M represents the node resource (memory) usage range.

In constructing the simulation regions, this section simulates specific edge scenarios of UAVs working together on the container cloud platform. Where each region is controlled by a controller, the number of edge server nodes and UAV devices in each region follows the actual distribution, and the number of test regions increases as the size of experimental UAVs increases.

In this section, using the average service failure recovery time and overall resource utilization rate as metrics, the system without high availability architecture and with randomly selected replacement UAVs as baseline is used for a detailed performance comparison and analysis with the system that uses service failure recovery time as a scheduling metric. The system uses resource utilization balance as a scheduling metric, and combines the two as scheduling metrics. However, in this paper we choose CPU and memory utilization as two metrics used to represent overall resource utilization rate. By measuring these two metrics on each node, we can calculate their variances respectively. Thus, we can determine overall resource utilization rate based on the equations introduced in Section 3.3.3.

5.2 Results

For the case of UAV failure as well as edge node failure, we launched tests separately by using average service failure recovery time, overall resource utilization rate metrics, and the fusion of the two as scheduling metrics. Finally, we simulated the two kinds of failure scheduling separately, and then verify that the proposed high availability mechanism meets the

scheduling expectations of the real scenarios.

5.2.1 Average service failure recovery time

We simulated offline failures of UAVs by randomly selecting UAVs and test the simulation on multiple experimental groups. As shown in Fig. 3, it demonstrates the results of service failure recovery time with the number of UAVs at the end of one simulation. The service failure recovery time of the system without high availability architecture and with randomly selected replacement UAVs gradually increases when the number of UAVs ranges within 300. This is because the baseline system relies heavily on randomness to replace UAVs by randomly selecting them when a failure situation is encountered, and the negative impact of randomness on service failure recovery increases significantly as the test area expands. When the number of UAVs increases to 500–700, the service failure recovery time decreases as the probability of picking up a neighboring UAV increases because the number of edge servers and UAVs in each region also increases. However, the performance is significantly inadequate when compared to several sets of experimental groups presented in this paper.

In this paper, we proposed an experimental group that dispatches experiments with expected service failure recovery time as the metric, an experimental group that dispatches experiments with expected overall resource utilization rate as the metric, and an experimental group that fuses the two. As can be seen

from Fig. 3, when the number of UAVs ranges within 100, the experimental group scheduling with expected service failure recovery time as an indicator can improve the system's failure recovery time to a certain extent in real-time situations by directly selecting the UAV with the shortest expected service failure recovery time for replacement. However, from the results of the subsequent experiments, greedy scheduling does not contribute much in terms of the system's stabilization and does not perform as well as the other two experimental groups as the scale increases.

For the experimental group where the projected resource utilization balance is the metric for scheduling the experimental group, the system can be ensured to remain in a steady state over a long period of time by choosing a scheduling scheme that has the minimum system resource utilization balance. However, when the number of UAVs ranges within 100, simply pursuing the most steady state will sacrifice the important metric of service failure recovery time. However, as the experimental scale increases, the system's homeostatic advantage is demonstrated.

The final proposed fusion solution combines the advantages of the first two sets of experiments, and is able to optimize the system's long time steady state and service failure recovery time at the same time as the experiment size changes. Compared with baseline, the fusion solution is able to reduce the service failure

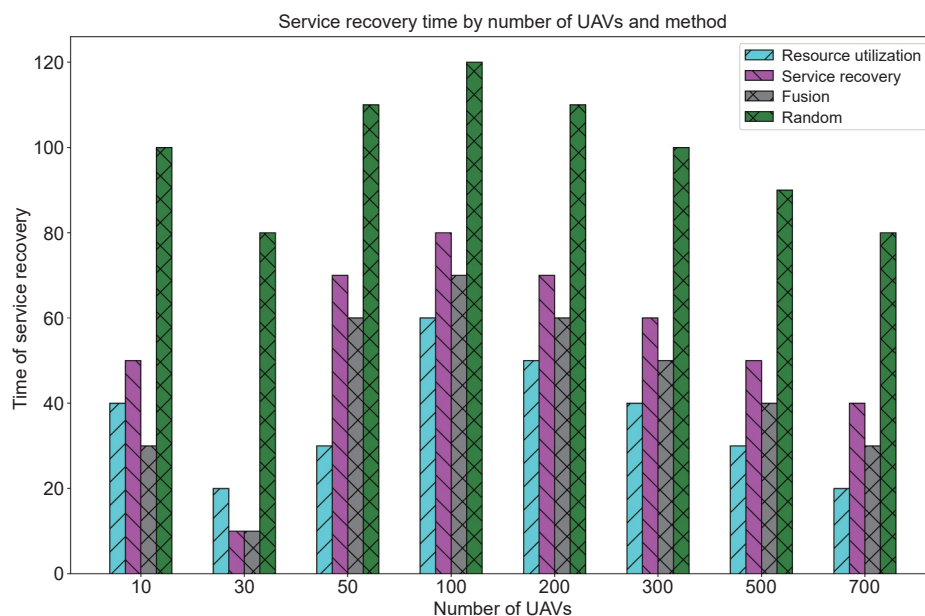


Fig. 3 Service failure recovery time as a function of the number of UAVs.

recovery time in the range of 12.37% to 68.94%, with an average reduction of 46.52%.

5.2.2 Balance of overall resource utilization

We conducted several simulation experiments in a dynamic simulation environment. Although there was a small numerical difference in the same set of data across simulations, this did not change the final performance comparison results of each scenario, demonstrating the impact of each scheduling solution on the node resource utilization of the UAV clusters. Therefore, in multiple simulation experiments, we choose the architecture as the number of controllers is 50, the number of nodes communicating with each controller is 10, and the number of UAVs controlled by each node is 10, and its one simulation process over a 24-hour period, to extract the impact of each scheduling solution on the difference in the resource usage of the nodes of the cluster as a whole. The obtained data are plotted as node CPU utilization variance change curves and node memory utilization change curves, as shown in Fig. 4.

5.2.3 Failure scheduling simulation

First of all, we found that the variance of CPU utilization rate and memory utilization rate was large in the first five hours of the simulation, indicating that the cluster's computational resources were unevenly used, this was because the simulation employs a random consumption of UAV's task resources, and at the beginning of the simulation each node's UAV was randomly deployed to the task, which led to a large variance in the initial cluster's CPUs resource utilization. Except for the random scheduling algorithm curve, the variance of CPU utilization and memory utilization rates in the cluster started to decrease after five hours. Among these, the curve for the scheduling

algorithm based on cluster resources decreased the fastest. This was because the scheduling algorithm collected information about the resource utilization of the nodes controlled by the various control servers of the cluster, and computed the variance of the CPU utilization of the nodes and the variance of the memory utilization at the time of the scheduling to be proportionally combined to select the UAV that minimized the variance of cluster resource utilization for scheduling.

The largest variance was in the random selection of UAVs for replacement method. Through the blue curve, it was found that the highest CPU utilization variance reached 500, and the memory utilization variance was as high as 400. This was followed by dispatching according to the smallest service recovery time. After that, the dispatch algorithm adopted the proportional combination of the service recovery time and the resource utilization variance scheduling. It was analyzed that the large variance of resource usage in random scheduling was mainly due to the fact that when a UAV was damaged and replaced, it was likely that the UAV with less node resource usage was used to replace it. At this time, the resource usage of idle nodes remained unchanged, while the resource usage of nodes with busy resource usage increased. This led to a more unbalanced resource usage among nodes. According to the resource usage variance for scheduling method, it was probable that the node UAV with larger resource usage than itself was selected for replacement, which increased the resource usage of the replaced node. The resource usage of the node that provided the UAV synchronized with the reduction of CPU and memory consumption due to the reduction of the controller UAV.

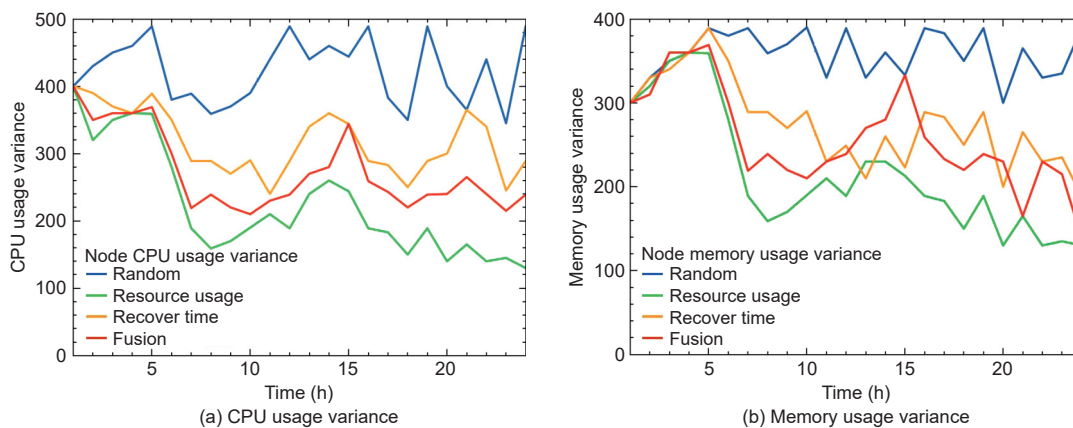


Fig. 4 CPU/Memory usage variance vs. time.

Furthermore, with the increase of simulation time, the resource variance of the cluster using resource variance for scheduling tended to stabilize. The green curve was almost unchanged after 20 hours because the resource utilization of each node was similar at this time, and no obvious resource utilization difference occurred when carrying out the replacement task. The fusion method adopted a proportional combination of the two scheduling methods of fault recovery scheduling and resource balancing scheduling, so that users could personalize the UAV failure scheduling according to their own requirements. The change of its variance could be observed through the red curve, and finally, the resource usage variance gradually decreased and tended to stabilize. The use of the method for scheduling not only ensured the speed of the failure recovery but also reduced the resource usage variance of the various nodes.

Several simulation experiments were conducted in a dynamic simulation environment. Although the same set of data had minor numerical differences across simulations, this did not change the final performance comparison results for each scenario. Therefore, in this section, we chose to extract the scheduling results during two simulations with the number of controllers as 7, the number of nodes as 343, and the number of UAV devices as 842, as shown in Fig. 5. The edge nodes and UAVs within the same circle are in the same region, and the connection relationship between the edge nodes and UAVs is represented by the dotted line.

According to Fig. 5, different experimental groups chose different scheduling outcomes based on their

respective scheduling based metrics. It can be seen that under large-scale experiments, the randomized experimental group relying on the negative impact caused by randomness does not meet expectations in scheduling result selection. And the experimental groups that rely solely on resource utilization balance or solely on service failure recovery time as metrics have certain limitations. The fusion high available solution proposed in this paper meets the scheduling expectations of real scenarios and has high practical significance. At the same time, the solution tries to avoid localized congestion situations, which can guarantee the system's steady state for a longer period of time.

6 Conclusion and Future Work

To address the problem of low resource utilization for UAV migration, we simulate a specific edge scenario of UAVs working together on a container cloud platform where we choose to use the UAV clustering framework proposed in this paper. We tend to optimize CPU utilization and memory utilization to maximize UAV resource utilization. To solve the application migration problem, we propose distributed scheduling algorithms and centralized scheduling algorithms according to different scenarios. Simulation results show that the proposed algorithms perform better than benchmark algorithms (e.g., greedy algorithms). We will consider extending this algorithm to multi-UAV scenarios in our subsequent work. Future research will explore how our framework can adapt to advancements in UAV and edge computing technologies, including

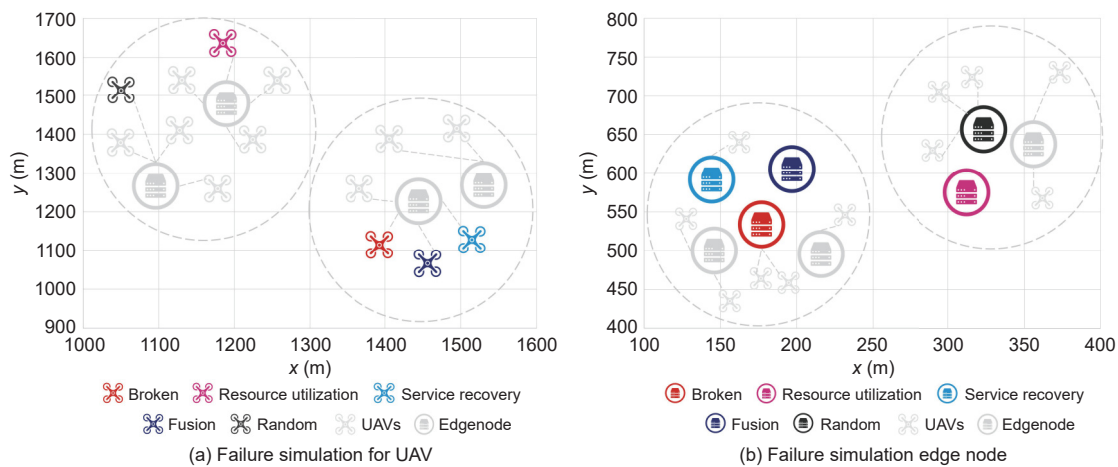


Fig. 5 Failure simulation for both circumstances. Red color represents the UAVs and edge nodes that fail during the simulation process, different colors represent the results of different dispatch strategy, and dashed circle represents an area of division.

integrating machine learning techniques for predictive maintenance and resource optimization, and incorporating edge AI for real-time decision-making and so on. What is more, we are looking forward to enhance the algorithm to increase the scheduling successful rate, since we did not consider the situation that migrate to wrong destination in this paper.

Acknowledgment

This work was supported in part by the National Key R&D Program of China (No. 2022YFF0902702), and in part by the Key R&D Program of Ningbo (No. 2023Z235), and in part by the Ningbo Yongjiang Talent Programme (No. 2023A-198-G), and in part by the Beijing Life Science Academy, Beijing, China (BLSA, No. 2023000CB0020), and in part by Henan Institute of Chinese Engineering Development Strategies, Zhengzhou, China (No. 2023HENZDB01).

References

- [1] J. D. Haan, Computing on the edge can be transformative—but look before you leap, <https://www.forbes.com/sites/forbestechcouncil/2021/03/15/computing-on-the-edge-can-be-transformative---but-look-before-you-leap/?sh=f061a826f3a5>, 2021.
- [2] C. Yang, W. Shen, and X. Wang, The internet of things in manufacturing: Key issues and potential applications, *IEEE Syst. Man Cybern. Mag.*, vol. 4, no. 1, pp. 6–15, 2018.
- [3] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, Are unmanned aircraft systems (UASs) the future of wildlife monitoring? A review of accomplishments and challenges, *Mamm. Rev.*, vol. 45, no. 4, pp. 239–252, 2015.
- [4] J. Zhao, W. Gao, Y. Wang, and G. Cao, Delay-constrained caching in cognitive radio networks, *IEEE Trans. Mob. Comput.*, vol. 15, no. 3, pp. 627–640, 2016.
- [5] W. Shi and S. Dustdar, The promise of edge computing, *Computer*, vol. 49, no. 5, pp. 78–81, 2016.
- [6] A. Moheddine, F. Patrone, and M. Marchese, UAV and IoT integration: A flying gateway, in *Proc. 26th IEEE Int. Conf. Electronics, Circuits and Systems (ICECS)*, Genoa, Italy, 2019, pp. 121–122.
- [7] O. Ghorbanzadeh, S. R. Meena, T. Blaschke, and J. Aryal, UAV-based slope failure detection using deep-learning convolutional neural networks, *Remote Sens.*, vol. 11, no. 17, p. 2046, 2019.
- [8] Y. Yazid, I. Ez-Zazi, A. Guerrero-González, A. El Oualkadi, and M. Arioua, UAV-enabled mobile edge-computing for IoT based on AI: A comprehensive review, *Drones*, vol. 5, no. 4, p. 148, 2021.
- [9] D. Erdos, A. Erdos, and S. E. Watkins, An experimental UAV system for search and rescue challenge, *IEEE Aerosp. Electron. Syst. Mag.*, vol. 28, no. 5, pp. 32–37, 2013.
- [10] K. J. S. White, E. Denney, M. D. Knudson, A. K. Mamerides, and D. P. Pezaros, A programmable SDN+NFV-based architecture for UAV telemetry monitoring, in *Proc. 14th IEEE Annu. Consumer Communications & Networking Conf. (CCNC)*, Las Vegas, NV, USA, 2017, pp. 522–527.
- [11] S. Zhu, L. Gui, N. Cheng, Q. Zhang, F. Sun, and X. Lang, UAV-enabled computation migration for complex missions: A reinforcement learning approach, *IET Commun.*, vol. 14, no. 15, pp. 2472–2480, 2020.
- [12] X. Z. Tian, P. T. Miao, and L. M. Zhang, Deep reinforcement learning for scheduling and offloading in UAV-assisted mobile edge networks, *Wirel. Commun. Mob. Comput.*, vol. 2023, p. 3940318, 2023.
- [13] X. Wang, S. Shi, and C. Wu, Research on service function chain embedding and migration algorithm for UAV IoT, *Drones*, vol. 8, no. 4, p. 117, 2024.
- [14] N. H. Motlagh, M. Bagaa, and T. Taleb, UAV-based IoT platform: A crowd surveillance use case, *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, 2017.
- [15] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, LSAR: Multi-UAV collaboration for search and rescue missions, *IEEE Access*, vol. 7, pp. 55817–55832, 2019.
- [16] F. Qi, X. Zhu, G. Mang, M. Kadoch, and W. Li, UAV network and IoT in the sky for future smart cities, *IEEE Network*, vol. 33, no. 2, pp. 96–101, 2019.
- [17] C. Dong, Y. Shen, and Y. Qu, A survey of UAV-based edge intelligent computing, (in Chinese), *Chin. J. Intell. Sci. Technol.*, vol. 2, no. 3, pp. 227–239, 2020.
- [18] P. McEnroe, S. Wang, and M. Liyanage, A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges, *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, 2022.
- [19] E. L. Lawler and D. E. Wood, Branch-and-bound methods: A survey, *Oper. Res.*, vol. 14, no. 4, pp. 699–719, 1966.
- [20] T. Westerlund and F. Pettersson, An extended cutting plane method for solving convex MINLP problems, *Comput. Chem. Eng.*, vol. 19Suppl1, pp. 131–136, 1995.
- [21] R. Bellman, Dynamic programming, *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [22] F. Glover and M. Laguna, Tabu search, in *Handbook of Combinatorial Optimization*, D. Z. Du and P. M. Pardalos, eds. New York, NY, USA: Springer, 1998, pp. 2093–2229.
- [23] D. Bertsimas and J. Tsitsiklis, Simulated annealing, *Stat. Sci.*, vol. 8, no. 1, pp. 10–15, 1993.
- [24] S. Mirjalili, *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Cham, Switzerland: Springer, 2019.
- [25] M. Dorigo, M. Birattari, and T. Stutzle, Ant colony optimization, *IEEE Computat. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006.
- [26] J. Kennedy and R. Eberhart, Particle swarm optimization, in *Proc. Int. Conf. Neural Networks*, Perth, Australia, 1995, pp. 1942–1948.
- [27] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, State-of-the-art in artificial

- neural network applications: A survey, *Heliyon*, vol. 4, no. 11, pp. e00938, 2018.
- [28] S. Marimuthu, S. G. Ponnambalam, and N. Jawahar, Threshold accepting and Ant-colony optimization algorithms for scheduling m -machine flow shops with lot streaming, *J. Mater. Process. Technol.*, vol. 209, no. 2, pp. 1026–1041, 2009.
- [29] M. Nejati, I. Mahdavi, R. Hassanzadeh, and N. Mahdavi-Amiri, Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint, *J. Ind. Prod. Eng.*, vol. 33, no. 7, pp. 459–471, 2016.
- [30] Y. Pan, S. Li, Q. Chen, N. Zhang, T. Cheng, Z. Li, B. Guo, Q. Han, and T. Zhu, Efficient schedule of energy-constrained UAV using crowdsourced buses in last-mile parcel delivery, *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 1, p. 28, 2021.
- [31] Y. Cui, D. Zhang, J. Zhang, T. Zhang, L. Cao, and L. Chen, Distributed task migration optimization in MEC by deep reinforcement learning strategy, in *Proc. 46th Conf. Local Computer Networks*, Edmonton, Canada, 2021, pp. 411–414.
- [32] Y. Ren, X. Chen, S. Guo, S. Guo, and A. Xiong, Blockchain-based VEC network trust management: A DRL algorithm for vehicular service offloading and migration, *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8148–8160, 2021.
- [33] C. Wang, J. Peng, F. Jiang, X. Zhang, W. Liu, X. Gu, and Z. Huang, An adaptive deep Q-learning service migration decision framework for connected vehicles, in *Proc. 2020 IEEE Int. Conf. Systems, Man, and Cybernetics*, Toronto, Canada, 2020, pp. 944–949.
- [34] M. Maray, E. Mustafa, J. Shuja, and M. Bilal, Dependent task offloading with deadline-aware scheduling in mobile edge networks, *Internet Things*, vol. 23, p. 100868, 2023.
- [35] M. Shakeri and A. Safaei, A review of the artificial intelligence methods in UAV guidance and control, *J. Unmanned Veh. Syst.*, vol. 7, no. 3, pp. 136–148, 2019.



Haoran Jin was born in Shenyang, Liaoning Province, China, in 2005. He is currently pursuing the BS degree in mechanical engineering, Zhejiang University, Jiaxing, China. His current research interests include edge computing, machine learning, and deep learning.



Chen Zhi received the PhD degree in computer science from Zhejiang University, Ningbo, China in 2022. He is currently a distinguished associate research fellow at School of Software Technology, Zhejiang University, Ningbo, China.



crowd computing.

Yi Li received the BEng degree in computer science and technology from Jiangnan University, Wuxi, China, in 2022. She is currently pursuing the master degree in software engineering at School of Software Technology, Zhejiang University, Ningbo, China. Her research interests include edge computing and



analysis, data-driven security, and AI and security.

Xuhong Zhang received the PhD degree in computer engineering from University of Central Florida, Orlando, FL, USA, in 2017. He is a ZJU 100-Young Professor with School of Software Technology at Zhejiang University, Ningbo, China. His research interests include distributed big data and AI systems, big data mining and



Shengye Pang received the BS degree in computer science from Tongji University, Shanghai, China in 2015. He is pursuing the PhD degree at College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His current research interests include service computing and trusted computing.



Yueshen Xu received the PhD degree from Zhejiang University, Hangzhou, China, 2016, and was a visiting scholar at University of Illinois at Chicago, Chicago, IL, USA. He is an associate professor at School of Computer Science and Technology, Xidian University, Xi'an, China. His research focuses on mutually-connected topics around multiple forms of services in typical computing/networking environment.



the computing and serverless systems.

Jinsong Shu received the BS degree in software engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2023. He is currently pursuing the master degree in software engineering at School of Software Technology, Zhejiang University, Ningbo, China. His research mainly focus on cloud



Junlan Cai received the master degree in food science from Zhengzhou Tobacco Research Institute of CNTC, Zhengzhou, China, in 2002. She is a professor of engineering at Beijing Life Science Academy, Beijing, China, and Zhengzhou Tobacco Research Institute of CNTC, Zhengzhou, China. Her current research

interests include tobacco and cigarette smoke chemistry, novel tobacco products and data science research of tobacco biological effects and biosynthesis.



Shaofeng Liu received the PhD degree from Fuzhou University, Fuzhou, China, in 2007. He is a professor with Beijing Life Science Academy, Beijing, China, and Zhengzhou Tobacco Research Institute of CNTC, Zhengzhou, China. His current research interests include tobacco and cigarette smoke chemistry, novel tobacco

products and data science research of tobacco biological effects and biosynthesis.



Xiang Li is a professor of engineering at Beijing Life Science Academy, Beijing, China, and Zhengzhou Tobacco Research Institute of CNTC, Zhengzhou, China. His current research interests include toxicological evaluation of tobacco smoke and biological effects of tobacco.



Xinkui Zhao received the PhD degree from Zhejiang University, Hangzhou, China, in 2016. He is a ZJU 100-Young professor with School of Software Technology, Zhejiang University, Ningbo, China. His research interests mainly include cloud-native technologies, edge computing, and service computing.