FlexEdge: Dynamic Task Scheduling for a UAV-based On-Demand Mobile Edge Server

Hui Sun, Bo Zhang, Xiuye Zhang, Ying Yu, Kewei Sha *Senior Member, IEEE*, and Weisong Shi, *Fellow, IEEE*

***Abstract*—With the large number of cameras deployed in smart industrial parks and smart campuses, edge devices and location-fixed edge servers are deployed near to these cameras and help transmit video streams to data center for video analytics; however, location-fixed edge servers are difficult to adapt to computation-intensive and delay-sensitive video analytics tasks in hot-spot, and it is also challenging to execute tasks in natural disasters in which the infrastructure is damaged. Moreover, task migration methods are used to balance the load of edge servers caused by irregular movement of detected objects, but it results in extra data transmission overhead. Therefore, unmanned aerial vehicles (UAVs) with computing and communication resources are widely used to optimize mobile edge video analysis; however, existing solutions formulate the UAV-based lowest latency and energy consumption by jointly optimizing the task allocation strategy and UAV location to be a multi-objective optimization problem, based on which the Pareto optimum solution set including task allocation strategies and UAV locations can find multiple solutions but not a unique solution. It makes the solution difficult to be applied in video analytics with UAV hover location decision-making scheme and task allocation strategy. In this paper, we propose a flexible cloud-edge collaborative scheduling strategy based on a UAV named FlexEdge. We first normalize values of execution time and energy consumption, and then convert the multi-objective optimization problem into a single-objective optimization problem by using the weighted sum of the two metrics as the optimization objective. We also proved the task allocation strategy based on execution time, energy consumption, and the UAV hover location decision-making scheme as an NP-hard problem. We propose a flexible and lightweight genetic algorithm (FGA) based on a polysomy-strengthening elitist genetic algorithm in FlexEdge to address the NP-hard problem. FlexEdge not only achieves optimal task allocation and UAV location to minimize the weighted sum of execution time and energy consumption, but also provides computing resources and reliable network connection to reduce task offloading overload, which is validated by comprehensive performance evaluation.**

***Index Terms*—Mobile edge computing, Unmanned aerial vehicles, Edge artificial intelligence, Intelligent video analytics, Edge computing**

I. Introduction

IN the era of the Internet of Everything, cameras [1] [2] are used to acquire video streams for video analysis in real-time. Mobile edge video analytics has been widely studied [3]–[5]. To meet real-time and reliability requirements, researchers designed task scheduling strategies in edge video analytics. Video analytics is mostly performed in the cloud, but high-definition video data transmission burdens the network [1]. The edge computing paradigm [6], [7] uses resources close to the data source to conduct video analytics, achieving low computing latency and energy consumption [8]–[10]. However, challenges remain in wired/wireless deployment and the low computing power of edge gateways in mobile edge environments. A location-fixed edge server hardly meets the demand of delay-sensitive video analytics. In an area with damaged infrastructure, a fixed server hardly provides sufficient computing resources for video analytics [11]. In addition, local computing resources cannot meet the performance requirements of video analysis tasks that unevenly distribute among edge devices. It is difficult for an edge device to continuously execute video analytics because of its limited resources (*e.g.,* battery) and harsh scenarios without available infrastructure (*e.g.,* natural disasters).

Given mobility, unmanned aerial vehicles (UAVs) with computing and communication resources are employed for mobile edge video analysis [12]–[16] in intelligent agriculture [17], natural disaster detection [18], emergency management [19], and intelligent transportation systems [20]. When a UAV-based edge server receives offloaded tasks from ground nodes, the trajectory is affected by the network between each ground node and the edge server [14]. It is challenging to optimize the UAV trajectory to utilize the wireless network to achieve the best channel quality. Although offloading data to mobile edge servers can improve latency [21]–[23], large-scale data transmission on the wireless network aggravates the network burden. It is challenging to allocate network resources on the UAV to meet communication requirements for data offloading. UAVs with a fully autonomous flight can use the bandwidth to transmit video streams [24], in which an adaptive video processing pipeline can be applied to achieve a dynamic task optimization of UAVs. A prior work studied task allocation strategies of UAVs in a mobile edge environment [15]. This study unveiled that the performance of video analytics is improved by the game computing offloading method, which allocates tasks to edge nodes or ground bases. A UAV-based task allocation framework was built according to its mobility [25]; and the UAV was used as a relay device. In prior work [16], a UAV-assisted edge computing system offers computing resources for many ground users. A portion of computing tasks are allocated to the UAV, and the remaining ones are processed

Hui Sun, Bo Zhang, Ying Yu, and Xiuye Zhang are with the School of Computer and Science, Anhui University, Hefei, China. E-mail: sunhui@ahu.edu.cn, e19301162@stu.ahu.edu.cn

Kewei Sha is with the department of Computer Science at University of Houston - Clear Lake (UHCL), Houston, TX, USA. E-mail: sha@uhcl.edu.

Weisong Shi is with Department of Computer Science, Wayne State University, Detroit, MI, USA. E-mail: weisong@wayne.edu

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

locally.

The above research aims to minimize the UAV-based system's latency and energy consumption by jointly optimizing the task allocation strategy and UAV hover location decision-making scheme. But, existing work mostly formulates execution time and energy consumption as a multi-objective optimization problem; however, it is difficult to find a unique solution from multiple ones of the Pareto optimum solution set [16] (*i.e.,* task allocation strategy and UAV hover location). We proposed a flexible cloud-edge collaborative scheduling strategy based on a UAV in a three-layer framework, **FlexEdge**. We first normalize execution time and energy consumption, and then convert the multi-objective optimization problem into a single-objective optimization problem by using the weighted sum of the two metrics as the optimization objective. We formulate the task allocation strategy and the UAV hover location decision-making scheme as an NP-hard problem. FlexEdge has a **f**lexible and lightweight **g**enetic **a**lgorithm (FGA) based on a polysomy-strengthening elitist genetic algorithm that addresses the NP-hard problem. FlexEdge aims to achieve efficient on-demand management of computing, power, and bandwidth resources at three layers. FlexEdge collaboratively conducts offloading tasks to lessen unbalanced tasks on fixed edge nodes in mobile edge environments.

Our contributions are summarized as follows.

- We propose an execution time model, an energy consumption model, and a bandwidth model for a three-layer framework on a UAV-based flexible air edge server (FES). When edge node locations are fixed, the UAV location is adjusted by periodically calculating the bandwidth and the cost of execution time and energy consumption. We can obtain the execution time and transmission latency of a task according to the bandwidth.
- We design a task allocation strategy based on execution time, energy consumption, and a UAV's dynamic location. Based on the location of edge devices, tasks, and transmission data, the problem of a task allocation strategy and UAV hover location decision-making scheme is converted into a single-objective optimization problem that minimizes the normalized weighted sum of execution time and energy consumption. The problem is proven to be an NP-hard problem in this study.
- We propose FGA for the UAV hover location decision-making scheme and task allocation strategy. Two chromosomal coding methods are employed to encode each device's task allocation strategy (discrete values), and UAV hover location (continuous quantity). The approximate global optimal solution can be obtained by iteratively performing individual fitness measurement, individual selection, operator crossover, and operator mutation.
- In addition, we study a UAV-based edge computing platform on a three-layer framework with edge nodes, an FES, and a cloud server in a real-world environment. We implement efficient management of execution time, bandwidth, energy consumption, and UAV hover location in four applications (*i.e.,* HAAR, deep neural network (DNN), Max margin object detection (MMOD), and tiny-YOLOv3) under Wi-Fi, 4G, and 5G networks. Results unveil that the performance of hybrid offloading approaches is significantly better than that of the cloud offloading approach. Our proposed FGA outperforms other hybrid offloading algorithms.

The rest of the paper is organized as follows. Section III explains our motivation to combine an FES with the cloud. We describe the system platform and key technologies in Section IV. The system model is established in Section V. Section VI proposes the solution to the cloud-edge collaboration problem, and the PSEGA-based algorithm (FGA) to solve this problem is proposed. In Section VII, we present the setting and devices in the experiment. Experimental results are analyzed in Section VIII. Related work is demonstrated in Section II. Finally, we draw the conclusion in Section IX.

II. Related Work

A. Mobile Server

UAVs with communication resources are mostly employed as relays in complex scenarios. Mohamed et al. [26] proposed a method to deploy UAVs in a fog computing framework to reduce network latency. Guillen et al. [27] optimized network connection for UAVs. Wang et al. [24] studied four strategies (EarlyDiscard, Just-in-Time-Learning, Reachback, and Context-Aware) to utilize bandwidth fully. Limited bandwidth can cause a final perceived latency and packet loss in a video stream. Molina et al. [28] proposed a framework to improve the adaptive video stream of UAVs in the network. The communication of multiple UAVs is unstable because the UAV frequently changes its hover location. Prior work studied a UAV communication network for a mobile ad-hoc networks (MANETs) [29] based on the conventional ad-hoc network [30], aiming to adapt to the distributed characteristics [9].

With a load increment, a UAV is equipped with communication and computing resources to process tasks. Kalatzis et al. [31] combined sensors and computing resources on a UAV with resources in a fog-computing model to detect forest fires. Zhan et al. [16] proposed a UAV-assisted mobile edge computing model to serve ground devices. Some tasks are allocated to the UAV while the remaining tasks are processed locally; however, computing resources in fixed servers are not fully utilized. Thus, we propose a scheduling strategy based on a three-layer framework, in which all communication and computing resources can be utilized to improve execution time, energy consumption, and bandwidth in UAV-based video analytics.

B. UAV Communication in Mobile Edge Computing

Research on UAV communication mostly focuses on the following three aspects. (1) Using a UAV as a relay node to provide a link for devices without the communication range of a fixed server [24], [26], [27]; (2) With its mobility, a UAV collects the source data in an area that humans cannot obtain [18], [20], [31]; (3) Information exchange among UAVs [11], [25], [32].

The UAV-based computing offloading method has been widely used because of the limited resources of mobile systems, such as the battery, network bandwidth, and storage

capacity. Kumar et al. [13] proposed methods to overcome these limitations by sending tasks to servers with sufficient resources. Wu et al. [33] studied the minimum throughput over all ground users in the downlink communication by optimizing the scheduling of multi-user communication, trajectory, and power control of a UAV. Block coordinate descent and successive convex optimization techniques were used to optimize objective parameters. Callegaro et al. [12] employed deep learning models to validate the feasibility of real-time processing for a video stream. A queue technology was used to establish a sensing-analysis-control framework in object-tracking applications. However, the queue structure only focuses on the bandwidth and feedback time. In this study, we established a bandwidth model for UAVs and ground nodes to address connection issues when these devices are out of communication range.

C. Computation Offloading and Path Optimization in MEC

In MEC, the computation offloading strategy is prevalent in UAV-based systems [15].

Offloading computation to the ground base station from edge nodes improves performance and balances execution time and energy consumption. Hu et al. [34] proposed a UAV-assisted MEC with computing resources to server ground users. A portion of tasks are allocated to the UAV; and the rest are processed locally. In a cycle, the sum of the maximum delay among users in each time slot is minimized by optimizing the UAV path, the ratio of allocation tasks, and scheduling variables. Rahman et al. [35] studied software-defined networking based on multi-UAV cooperation to maximize throughput and allocated tasks. Ma et al. [36], [37] designed a mobile edge server-based cluster to propose a task allocation strategy and communication for mobile users. Liu et al. [25] studied the computation offloading and routing optimization problems for UAVs under the UAV-edge-cloud architecture. Tasks on the UAV cluster are migrated to edge servers or the cloud. UAVs are used as edge devices. A mobile edge collaboration network of a UAV [38] is proposed to optimize the communication between the UAV and ground nodes, maximizing the long-term availability of the UAV in remote areas that lack computing services. A Markov process and deep reinforcement learning model were used for offloading decision and resources management.

Chen et al. [39] studied the integration of wireless communication and multi-access edge computing. Researchers formulate the task offloading problem as a multi-agent Markov decision process to address uncertainty and limited resource sharing issues. They also develop a distributed learning framework beyond the 5G network and an online distributed reinforcement learning algorithm; however, this work rarely considers mobile edge servers. They also focus on an air-ground multi-access edge computing system [40]. A non-cooperative stochastic game model based on mobile users is proposed to maximize its expected long-term payoff; afterward, this model is converted to a single-agent Markov decision process. Using reinforcement learning (RL), the decision is made without history knowledge in a vast state space composed of all mobile users (MUs). We focus on efficient resources utilization, real-time task processing, and task allocation in an area without required infrastructure.

In a mobile edge environment, studies on UAVs primarily include (1) UAV path optimization, (2) using UAVs as a data acquisition device in MEC [15], and (3) task allocation at ground edge servers through limited bandwidth [11]. In addition, a UAV equipped with computing resources is applied to establish a mobile edge server [25]. Prior work designed a task allocation strategy to optimize execution time and energy consumption for resource-constrained ground nodes and UAVs. In this work, we propose a FES to minimize the weighted sum of execution time and energy consumption.

III. Motivation

A. Dynamic Density of Tasks

Many cameras are deployed to detect dynamic crowd density and gathering locations in smart industrial parks and smart campuses. These cameras mostly connect to a data center through gateways. However, due to the difficulty of wiring and insufficient computing power at the gateway, it is challenging to meet the demands of computing-intensive and delay-sensitive tasks in a mobile edge environment. For example, tourists move to the exit from the entrance in a smart park. There are obvious aggregations and dispersions in each spot; meanwhile, the location and number of tourists change frequently. A high crowd density produces many video analytics for video streams from cameras (*e.g.,* crowd density detection, object detection, and recognition). In addition, the irregular movement of detected objects and an uneven distribution of tasks in an area lead to changes in video analytics and data volume at an edge node. Mobile edge computing is a promising method for optimizing the latency and energy consumption of video analytics [41]–[43].

B. Prompt Deployment of UAVs with Edge Servers

UAVs have been widely used in agricultural production [17], natural disaster detection [18], and emergency management [19]. A UAV-based edge server provides computing resources close to the data source on edge nodes in a mobile edge environment.

Fixed edge nodes are used for video analysis. However, due to many devices, complex wiring, and large-scale video streams, they dramatically consume computing resources and bandwidth rather than receiving feedback in real-time. UAVs provide mobile computing resources for edge nodes. Studies on UAVs include integrating hybrid computing resources in a multi-layer framework, building UAV-based mobile serves, and allocating tasks to the server based on the wireless signal path loss [11]. However, they cannot fully use the UAV's computing power, mobility, communication, and multi-layer resources.

As shown in Fig. 1, the location and wiring of ground edge devices are complex. The low powerful-computing edge gateways in wired networks face challenges. It is difficult for a location-fixed edge server to meet the requirement of latency for uneven video analysis tasks. In addition, it is hard to guarantee the network connection and available

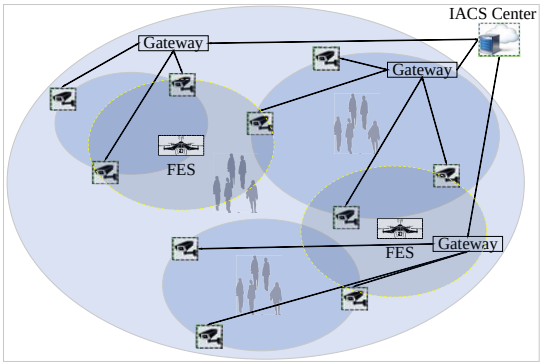


Fig. 1: Each edge node connects to the IACS center (*i.e.,* the Institute of Advanced Computer Systems) under a wireless network.

resources for video analytics by using a fixed server in an area with disrupted infrastructure. UAVs can fly to a location, maximizing system resources utilization based on the location of edge nodes. It must be noted that geographical constraints should be considered to avoid the UAV from crashing into buildings in realistic scenarios.

In this work, our proposed UAV-based FlexEdge can dynamically adjust the UAV location to allocate multi-layer resources. We propose a collaborative scheduling scheme to reduce execution delay and energy consumption in computing-intensive and latency-sensitive video analytics. Our method aims to improve the utilization of computing resources for video analysis at multiple layers in a mobile edge environment.

IV. Overview System and Implementation

We proposed a flexible cloud-edge collaborative scheduling strategy in a three-layer framework named **FlexEdge** for mobile edge video analytics, see Fig. 2. We design a task allocation strategy and UAV hover location decision-making scheme to reduce execution time and energy consumption in computing-intensive and latency-sensitive video analytics.

A. Overview System

In Fig. 2, the three-layer framework is composed of edge nodes, a UAV-based flexible air edge server (FES), and a cloud server. Communication resources include a wireless network between edge nodes and the FES and three types of networks (*e.g.,* Wi-Fi, 4G, and 5G) between edge nodes and the cloud. The FES cannot directly connect to the cloud server. Results are returned to edge nodes when video analytics on the FES completes. To fully utilize three-layer resources, an edge node can perform computing tasks locally or allocate them to an upper layer (*e.g.,* the FES or the cloud server) having enough computing resources. The FES makes the system adaptable and fully utilizes resources. Thus, a task allocation strategy is important in this system. The network between the edge nodes and the FES should be optimized for video transmission. Thus, we design a UAV hover location decision-making scheme to determine the optimized location, which improves the system's network bandwidth and resource utilization.

B. Fundamental Design

In an edge environment, video analysis can extremely burden the edge devices, while data transmission can lead to data congestion and network overload. These tasks are currently handled by local devices and cloud servers, which costs time, bandwidth, and battery.

Our proposed FlexEdge (see Fig. 2) fully utilizes resources of ground edge nodes, an FES, and a cloud server to analyze video tasks [1]. An edge node is composed of a camera and a development board, such as Raspberry Pi 3/4, NanoPC-T4, and Jetson Nano which provide computing capability at edge nodes. The FES is equipped with Manifold2-G and wireless routers. The cloud server connects to a wireless base station. We design three video stream channels, including a local data channel at the edge node, a wireless network data channel from the edge node to the cloud server, and a wireless network data channel from the edge node to the FES.

An edge node requests the task allocation strategy from the cloud server to perform local computation or computation offloading model. In local computing, video is processed at the edge node; afterward, execution time and energy consumption are recorded. There is no execution time and energy consumption for data transmission in local computing. In computing offloading, local tasks are offloaded to the FES or the cloud server. Video is processed (*i.e.,* video extraction, encoding, and compression) at an edge node; and, it is transmitted to the upper layer through the data channel. The execution node returns detection results. The total time (or energy consumption) is spent on execution and transmission at the three layers.

The FES conducts video analytics using computing resources in on-board Manifold2-G, once it receives video data from an edge node. The UAV hover location, execution status, and video analytics tasks from the ground nodes are sent to the cloud server, which conducts our proposed FGA. We can obtain the task allocation strategy of each device and the UAV hover location. The UAV's OSDK (on-board SDK) is used to control its movement to the hover location. OSDK is a development toolkit for developing applications, which runs on the Manifold2-G. Developers can obtain the information from the UAV by calling the interface specified in OSDK. According to developers' software logic and algorithm framework, users control the UAV to perform actions, *e.g.,* automated flight. We focus on the task execution and transmission energy consumption of the FES. Meanwhile, we ignore the flying energy consumption in this work [44]–[47]. In this system, the FES cannot connect to the cloud server. The allocation strategy and the UAV hover location are sent to edge nodes; meanwhile, the FES receives the allocation strategy from its connected edge node.

The cloud server collects status and tasks at edge nodes and the FES. Then, the task allocation strategy and the UAV hover location that are made by our proposed FGA are sent to corresponding nodes. The UAV hover location is mainly determined by the current UAV location, task information,

[1]In this work, video analytics includes face detection and object detection applications.

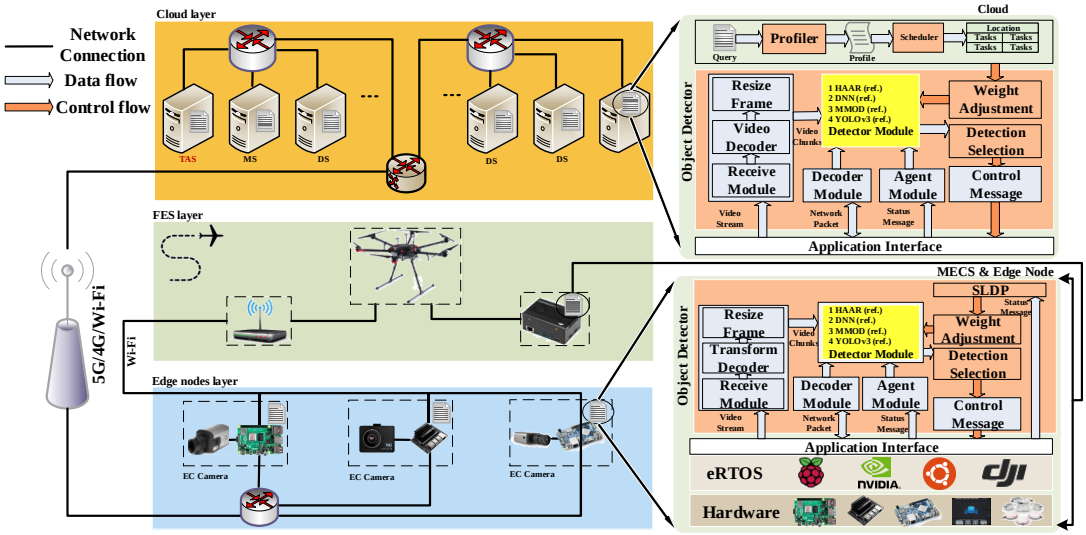


Fig. 2: The Overview System of the UAV-based FlexEdge. The system is divided into three layers according to the computing power and network (*i.e.,* edge nodes layer, a FES layer, and a cloud server layer). The software framework designed for video analytics tasks is distributed in edge nodes, a FES, and a cloud server. The TAS is a task allocation server in the cloud.

and wireless bandwidth. First, the UAV signal coverage and bandwidth are calculated based on the Euclidean distance, path loss, and Shannon theorem in the three-dimensional space. The channel has little impact on the bandwidth, and we ignore the bandwidth change caused by nodes' connection to the router in the system. Then, we normalized the minimum weighted sum of execution time and energy consumption for video analytics and data transmission. We propose FGA to obtain an approximate optimal objection. The FGA encodes the spatial coordinates of the UAV as continuous quantities and the offloading markers as discrete values. The recombination and mutation tiles are stored in a list. In evolution, each chromosome matrix can reorganize and mutate independently with the recombination and mutation operators in the list. The iteration performs the following steps: evaluation of individual fitness of the population, selection, crossover, and variation, and searches for the approximate global optimal solution. Then, the cloud server can change the detection procedure according to the application. Finally, the cloud server records the device status, network condition, execution time, and energy consumption in a log file.

V. System Models

In a mobile edge environment, we built models to study flexible task allocation services for video analytics. We proposed bandwidth, execution time, and energy consumption models in FlexEdge. Notations and definitions used in this paper are listed in Table I.

A. Bandwidth Model

Without a loss of generality, ground edge node k is denoted as q_k ($q_k \in Q$). We have $k = 1, 2, 3, ..., K, K = 8$ in this work. $(x_k,\ y_k,\ h_k)$ is the space vector coordinate of edge

TABLE I: Notations and Definition

Notations	Definition
Q	Set of edge devices, and Q=$\|Q\|$
M	Set of tasks, and M=$\|M\|$
q_k	The k-th edge node, k=1,2,3,...K
m_n	The n-th task, n=1,2,3,...N
N	The number of tasks in the system
(x_k, y_k, h_k)	The space vector coordinate of edge device q_k
(v_x, v_y, v_h)	The space vector coordinate of the UAV
d_k	Distance between edge device q_k and UAV
β_0	Average channel power gain at 1 m
γ_k	Reference SNR of edge device q_k at a distance of 1 m
σ^2	Noise power
θ	Path-loss exponent
R_0	Signal bandwidth (Hz)
P_k	The transmission power of edge device q_k
L	Local computing mode
F	FES computing mode
C	Cloud computing mode
d_n	Date transmission volume of task m_n
c_n	computation workload of task m_n
R_k^W	Data transmission rate depends on the distance between FES and edge device q_k
o_n	Task allocations indicator variable of task m_n
f	Computing capacity of devices
e	energy consumption per task unit
net	Candidate networks including 5G,4G, and Wi-Fi
Ψ	Computation workload of total tasks
T_C	the set of computation workload of all tasks
T_D	the set of data transmission volume of all tasks
SC	Amount of the total cost
S_C	Total cost of tasks at cloud layer
S_F	Total cost of tasks at FES layer
S_L	Total cost of tasks at local layer
G_n	Total cost for task n
α	Weigh of execution time in cost

node k. An edge node's h_k is a fixed value. The UAV location is denoted as $v = (v_x, v_y, v_h)$. The UAV can obtain edge nodes location that will not change in a cycle. The UAV hover location can be calculated based on task allocation and network flow, which decides the hover location in the next cycle. Then, we have

$$d_k = ||q_k - v|| = \left((x_k - v_x)^2 + (y_k - v_y)^2 + (h_k - v_h)^2\right)^{\frac{1}{2}}, \quad (1)$$

which represents the Euclidean norm. The 3D space distance between edge node k and the UAV is computed by Eq. (1). In this work, we consider the case that the channel between each device and the UAV is a line of sight path [48], [49]. Thus, the channel gain is

$$g_k = \beta_0 \cdot d_k^{-\theta} = \frac{\beta_0}{\left((x_k - v_x)^2 + (y_k - v_y)^2 + (h_k - v_h)^2\right)^{\frac{\theta}{2}}}, \quad (2)$$

where g_k is the channel power gain from edge node q_k to the FES. β_0 is the average channel power gain at one meter. θ is the path-loss exponent. According to Shannon theorem, the data transmission rate between q_k and FES in the Wi-Fi network is denoted as

$$R_k^W = R_0 \cdot \log_2\left(1 + \frac{P_k \cdot g_k}{\sigma^2}\right), \quad (3)$$

in which σ^2 is the noise power at the FES receiver. $P_k(k = 1, 2, 3, ...8)$ is the transmission energy at edge node q_k, W represents Wi-Fi and R_0 is the channel bandwidth. According to definition of the signal-to-noise ratio in communication, we have

$$\gamma_k \triangleq \left(P_k \beta_0 / \sigma^2\right) \quad (4)$$

which is the signal-to-noise ratio at one meter. Combining Eqs. (2), (3), and (4), we have

$$R_k^W = R_0 \cdot \log_2\left(1 + \frac{\gamma_k}{\left((x_k - v_x)^2 + (y_k - v_y)^2 + (h_k - v_h)^2\right)^{\frac{\theta}{2}}}\right) \quad (5)$$

B. Execution Time and Energy Consumption Models

Each edge has one task at least. Expression (c_n, d_n) represents the number of task c_n and transmission data volume d_n of task $m_n (n \in [1, N], m \in [1, M])$. The edge node can select a task allocation layer (local, FES, or cloud server). We do not consider the bandwidth cost of returned results because the feedback size is significantly smaller than video data [22], [49]–[51].

Local layer. There is no task offloading between an edge node and upper-layer servers in the local computing model. Thus, we only consider the execution time and energy consumption of a task. For task $m_n = (c_n, d_n)$, the rate of computing capacity at the edge node k^{th} is expressed as f_k^L. The execution time is shown as

$$T_{k,n}^L = \frac{c_n}{f_k^L}. \quad (6)$$

The energy consumption can be obtained as

$$E_{k,n}^L = c_n \cdot e_k^L, \quad (7)$$

in which e_k^L represents the energy consumption per computing cycle.

FES layer. The computing task at the edge node is allocated to the FES through Wi-Fi. The location of an FES changes when the current tasks complete in a cycle. Herein, the distance between the FES and an edge node primarily affects the bandwidth between the two nodes. The transmission delay is important for video analysis compared with the execution delay because the transmission time cost varies significantly with the distance between the edge node and the FES. Edge video analytics has strict real-time requirements. The execution time is denoted as

$$T_{k,n}^F = \frac{c_n}{f_k^F} + \frac{d_n}{R_k^W}, \quad (8)$$

where f_k^F is the rate of the computing capacity at the FES. The f_k^F is approximately an order of magnitude higher than that of the edge node on an average. The symbol R_k^W (see Eq. (3)) is the data transfer rate between edge node q_k and the FES under the wireless network. In this layer, the energy consumption is expressed as

$$E_{k,n}^F = c_n \cdot e_k^{Comp,F} + d_n \cdot e_k^{W,F}, \quad (9)$$

where $e_k^{Comp,F}$ represents the energy consumption per computing unit of a task from edge q_k on the FES. $Comp$ refers to as the task computing while F denotes the FES layer. The $e_k^{W,F}$ is energy consumption per unit data under the Wi-Fi network.

Cloud layer. Tasks can be allocated to the cloud through Wi-Fi, 4G, and 5G networks. The execution time on the cloud and transmission time is represented as

$$T_{k,n}^C = \frac{c_n}{f_k^C} + \frac{d_n}{R_k^{net,C}}, net \in \{5G, 4G, Wi-Fi\}. \quad (10)$$

The energy consumption of execution on the cloud and data transmission can be listed as

$$E_{k,n}^C = c_n \cdot e_k^{Comp,C} + d_n \cdot e_k^{net,C}, net \in \{5G, 4G, Wi-Fi\}, \quad (11)$$

where f_k^C is the frequency of computing capacity in the cloud. $R_k^{net,C}$ is the effective transmission rate between edge node q_k and the cloud. The $nets$ belongs to 5G, 4G, and Wi-Fi, and C is the cloud layer. $e_k^{Comp,C}$, which is the energy consumption per computing unit in the cloud, is set to zero because the cloud has sufficient power. We consider the edge node's transmission energy consumption. $e_k^{net,C}$ is the energy consumption per unit data transmitted to the cloud.

VI. Objective Optimization Problem

We proposed a task allocation strategy (based on the models of execution time, bandwidth, energy consumption) and UAV hover location decision-making scheme. We prove the problem is NP-hard, and we design FGA to obtain the task allocation strategy and the UAV hover location.

A. Problem Formation

Our work aims to minimize execution time and energy consumption for video analytics. When all tasks are allocated to the cloud server, the server hardly completes all tasks in real-time owing to the limitation of bandwidth and battery at an edge node. Otherwise, allocating tasks to all devices wastes computing resources in the cloud server. Thus, edge nodes need to choose an optimal allocation strategy (local, FES, and cloud) on-demand according to the execution time, energy

consumption, and bandwidth cost. The energy consumption of a UAV and edge nodes impacts the performance. The latency of computing-intensive tasks in an edge environment significantly determines the performance. Thus, we consider the execution time and energy consumption of computing and transmission.

We first normalize the execution time and energy consumption, and the features are transformed by scaling each feature to a given range. Then, different data features have comparable scales in terms of data units and data distribution. The normalization formula is given by

$$\begin{aligned} T_{std} &= \frac{T_{src} - T_{\min}(\text{ axis } = 0)}{T_{\max}(\text{ axis } = 0) - T_{\min}(\text{ axis } = 0)} \\ T_{\text{scaled}} &= T_{\text{std}} \cdot (\max - \min) + \min \\ T &= T_{\text{scaled}} \end{aligned} \tag{12}$$

where T_{src} represents the original data set of execution time. T_{min} and T_{max} are the minimum and maximum values in the original data set. Symbol $[min, max]$ is the feature range in which T_{src} is scaled. T is the normalized value of T_{src}. We used the same method to normalize energy consumption.

Then, we set a global cost formula to combine the two factors with weights as

$$SC = \sum_{n=1}^{n} \{\alpha \cdot T_n + (1-\alpha) \cdot E_n\}, n \in [1, N] \tag{13}$$

where SC is the total cost. α and (1 - α) are weights of execution time and energy consumption, respectively. Each weight can be dynamically configured in video analytics. When battery of the UAV is low, video analysis must complete as soon as possible, aiming to guarantee the UAV return to the base station before the battery is exhausted.

To minimize the cost of task computing, the geographical constraint-based objective optimization problem can be expressed as

$$\min_{(Q,M)} \sum_{n=1}^{N} \{\alpha \cdot T_n + (1-\alpha) \cdot E_n\}, n \in [1, N] \tag{14}$$

$$s.t. \quad \forall o_k \in \{0, 1, 2\}, \forall k \in K \tag{15}$$

$$c_n \in T_C, d_n \in T_D, n \in [1, N] \tag{16}$$

$$G_n = \alpha \cdot T_n + (1-\alpha) \cdot E_n, n \in [1, N] \tag{17}$$

$$S_L = \left\{ \sum c_n | o_n = 0 \right\}, n \in [1, N] \tag{18}$$

$$S_F = \left\{ \sum c_n | o_n = 2 \right\}, n \in [1, N] \tag{19}$$

$$S_C = \left\{ \sum c_n | o_n = 1 \right\}, n \in [1, N] \tag{20}$$

$$\sum_{x \in \{L,F,C\}} S_x = \Psi \tag{21}$$

$$\frac{d_n}{R_k} + \frac{c_n}{f_k} \leq max\{O_L, O_F, O_C\}, \forall k \in [1, K] \tag{22}$$

$$\frac{c_n}{f_k} + \frac{d_n}{R_k} \leq T_{max} \tag{23}$$

$$\sum_{n=1}^{N} T_{k,n}^{F} \cdot e_F + \sum_{n=1}^{N} E_{k,n}^{F-pre} \leq E_{Total}^{F} \tag{24}$$

Symbol G_n is the weighted sum of execution time and energy consumption for task *n*. S_L, S_F, and S_C are the weighted sum of two metrics for all tasks at local, FES, and cloud layers, respectively. The optimization objective of the problem is to minimize the normalized weighted sum of execution time and energy consumption in Eq. (14). It must be noted that task execution delay is employed as a part of the optimization objective in the objective optimization problem. We use Eq. (23) as a constraint on task execution delay (*e.g.,* delay threshold) to formulate the optimization problem.

We ignore the movement time of the UAV because the speed of UAV is higher than 30 m/s, which is smaller than the task execution time [52].

The linear weighted sum method, interactive methods, and ε-constraint methods are all popular multi-objective optimization models. The linear weighted sum method [15], [46], [53], [54] integrates all multiple objectives according to their weights. Interactive methods [55] consider the interrelationships among objective functions, and the evaluation formulas are designed to integrate optimization for multiple objective functions. The ε-constrained method [56] finds the interrelationship among objectives and selects one objective functions as the objective, setting the rest as constraints. We can obtain the intersection of the optimal solution sets from three optimization models (linear weighted sum method, ε-constrained, and Pareto) [57]. These models reduce the dimension of the multiple objectives, after which the multi-objective optimization problem is converted into a single-objective optimization problem. Pareto and linear weighted sum models are combined to find the optimal solution for a combination problem in service computing systems [53]. However, this combination is impractical for applications in the edge computing environment owing to the long computation time required to obtain the Pareto solution, which violates the real-time requirement of the edge task processing. However, a linear weighted sum model can address the problem in real-time. In addition, we compare our linear weighted sum method with the Pareto method in terms of the solutions (including task allocation strategy and UAV location) and validate that the task allocation strategy generated by our proposed method is contained in the Pareto solution set.

We normalize execution time and energy consumption to a range of the same size. Each data feature is scaled using the min-max function, one of the popular methods for pre-processing data in machine learning [58]. We convert the multi-objective optimization problem of execution time and energy consumption into a single-objective optimization problem by means of normalization. Our proposed FGA correctly dominates the objective function based on the unit and data distribution in the data set, including execution time and energy consumption. By executing the iterative in FGA, the result of each iteration correctly optimizes the objective function as expected.

The reason that we do not employ multi-objective optimization is described as follows. We can obtain a Pareto optimal solution set by employing a multi-objective optimization method based on execution time and energy consumption [16]. This solution set contains multiple optimal solutions (*i.e.,* task allocation strategies and UAV hover location). In this case, we cannot find a unique optimal solution. Thus, we adopt a single-objective optimization approach using a normalization method based on execution time and energy consumption, with the weighted sum of the two metrics as the optimization objective.

We propose FGA to find the optimal task allocation strategy and UAV hover location to minimize the weighted sum. The single-objective optimization method avoids the problem in multi-objective optimization. In computing resources-limited mobile edge environments, the single-objective optimization method is easier to achieve real-time goals with lower energy consumption in video analytics than multi-objective optimization under the requirement of accuracy. Our proposed method has good scalability and can meet different demands on execution time and energy consumption by setting weights of the two metrics in applications.

In a task-offloading cycle, our proposed method determines the optimal task allocation strategy and the following UAV hover location based on edge nodes' status, the number of tasks, data volume, computing resources, and communication resources. When the UAV moves to the target location, task offloading is performed. The UAV moves between two points in a task-offloading cycle. The process to determine the UAV hover location in different cycles is independent. The above operation is repeated to obtain the following UAV hover location. In this work, we did not employ FES to individually communicate with each edge node for the offloading task on edge nodes.

The input parameters of FGA include the locations, tasks, and data volume of all ground edge nodes, computing resources, and battery of nodes in the system. The output parameters are the task allocation strategy of edge nodes and the UAV hover location.

Eq. (15) shows the allocation indicator variables for each task. A task can select the edge node (*i.e.,* o_n= 0), the cloud server (*i.e.,* o_n=1), and the FES (*i.e.,* o_n=2) for video analytics. Eq. (16) denotes the set of computation workload and data transmission volume of all tasks in FlexEdge. Eq. (17) implies that the sum of weights of execution time and energy consumption is one. The weighted values of the two factors vary with different applications. Constraints in Eqs. (18), (19), and (20) denote the sum of tasks offloaded by the three layers, respectively. Eq. (21) indicates that computation workload of all tasks in each layer must be completed in an execution cycle. Eq. (22) means that a single-task execution time should not exceed the maximum value in the three layers.

To the best of our knowledge, there have been few studies conducted on real-world UAV platform-based task offloading and UAV hovering location optimization. In our work, we find that the energy consumption of rotary-wing UAVs is two orders of magnitude larger than that of task execution on our real-world mobile edge platform in this study. We employed flying time and energy consumption as constraints for task execution on the UAVs. The constraints of the flying time and energy consumption in the optimization problem are represented by Eqs. (23) (24). These two metrics were not considered as the optimization objectives in Eq. (14) because our work primarily aims to validate the task offloading strategy and UAV hover location on a UAV-based real-world mobile edge server. The energy consumption of task execution on UAVs is significantly lower than the consumption of energy during flying in realistic scenarios. The energy consumption of tasks plays an important role in the allocation strategies for a real-world platform.

Our work is similar to many prior works that focus on task allocation and ignore the flying time and flying energy consumption [15], [34], [38], [44]–[47], [59]–[62]. Thus, we focus on the task execution and transmission energy consumption of the FES, and ignore the flying energy consumption in this study.

It must be noted that different types of UAVs have various flight speeds and energy consumption. Delivery drones have a long flight time, and high flight energy consumption is the primary source of overhead in the system. They can deliver our UAVs that perform computational tasks to the desired locations, but we consider offloading task computation rather than the transport cost in this work. In addition, there is a lack of energy consumption for UAVs, with wireless power transfer [47] being a promising technology to improve battery life and extend computing time.

Eq. (25) means the cost at each layer.

$$G_{layer} = \begin{cases} \alpha \cdot T_L + (1-\alpha) \cdot E_L, & \text{if } o_n = 0, \\ \alpha \cdot T_C + (1-\alpha) \cdot E_C, & \text{if } o_n = 1, \\ \alpha \cdot T_F + (1-\alpha) \cdot E_F, & \text{if } o_n = 2. \end{cases} \tag{25}$$

We prove the objective optimization problem (Eqs. (14)∼(24)) in this section is a NP-hard problem in Appendix A.

B. FGA Task Allocation Strategy

Genetic algorithms [63], [64] are used to find the optimal solutions to a given computational problem that maximizes (or minimizes) a particular function. Genetic algorithms are much efficient than random search and exhaustive search algorithms [64], and they need not require extra information about the given problem. Genetic algorithms can find solutions to problems while other optimization methods cannot handle them when there is a lack of continuity, derivatives, linearity, or other features.

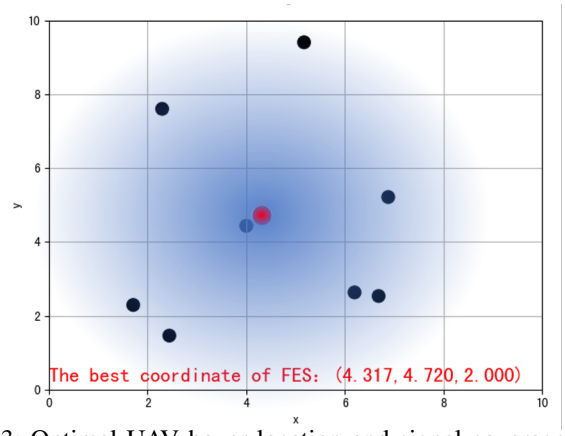


Fig. 3: Optimal UAV hover location and signal coverage area obtained by FGA

Feature	Continuous	Continuous	Discrete	Discrete	Discrete	Discrete
Value	x	y	Marker 1	Marker 2	...	MarkerN

0	Offload to Local
1	Offload to Cloud
2	Offload to FES

Fig. 4: Encoding methods for the task allocation strategy and the UAV hover location.

In the heuristic algorithm, the genetic algorithm is proposed to deal with structural objects.

To reduce the execution time and energy consumption, we design FGA to obtain an approximate optimal task allocation strategy for all edge nodes and the UAV hover location. For example, all tasks at an edge node are offloaded to the FES, aiming to obtain the optimal UAV hover location, see Fig. 3. Black dots represent the edge nodes within the UAV's wireless signal coverage. The red dot is the optimal UAV hover location determined by the transfer data volume, tasks, bandwidth, computing resources, battery of all nodes, and three-dimensional space distance between the edge nodes and the UAV in the system.

In this work, FGA is deployed on the cloud rather than the UAV. The UAV cannot connect to all edge nodes continuously due to its limited wireless communication range. Each ground edge node connects to the cloud server by the wireless network. An edge node transmits its status (*e.g.,* task completion or not) to the cloud in real-time. In addition, the real-time decision-making process in FGA consumes computing resources and energe. The FES is resource-constrained; thus, we deploy FGA in the cloud to fully utilize resources on the FES to reduce resources cost for task allocation strategy.

We used two chromosome-coding methods to determine the task allocation strategy and the UAV hover location. Discrete real numbers (*e.g.,* 0, 1, and 2) represent the allocation layer for each device. Continuous real numbers within the $[0, 10]$ represent the three-dimensional geometric coordinate $(x_k,\ y_k,\ h_k)$ of the FES. Eq. (13) is used to evaluate the fitness and operate on the population.

Algorithm 1 represents an iteration of a task allocation strategy and UAV hover location decision-making scheme in FGA. The algorithm obtains the location of edge nodes and the FES. We can set the evolution number and population number. Afterward, we initialize the population of Num individuals according to the coding rules (Lines $1 \sim 3$). Our proposed FGA is based on a genetic algorithm. We set the chromosome encoding method of each individual in the population according to the objective optimization function.

Algorithm 1 **F**lexible and lightweight **g**enetic **a**lgorithm (FGA) based on polysomy-strengthening elitist genetic algorithm in FlexEdge (FGA)

Input: A set $\{[x_k, y_k, h_k], \forall k \in K\}$:location of each edge node, a set of computation $\{(c_n, d_n), \forall n \in N, c_n \in T_C, d_n \in T_D\}$, and system budget Ψ.

Output: a set of tuples $\chi = \{((v_x, v_y, v_h), o_1, o_2 \cdots o_n), n \in [1, N], k \in [1, K]\}$, number of all cycles T.

Initialization: $I \leftarrow 200$, discretized grid γ with locations $\{[x_k, y_k, h_k], \forall k \in K\}$,$K' \leftarrow K$, $N' \leftarrow N$, tasks computation volume $\{c_n, \forall c_n \in T_C\}$,data volume $\{d_n, \forall d_n \in T_D\}$,

1: Initial population chromosome matrix;
2: Calculate Cost: $SC = \sum_{n=1}^{n} \{\alpha \cdot T_n + (1-\alpha) \cdot E_n\}, n \in N$;
3: Calculate Fitness;
4: **for** $t = 1, \cdots T$ **do**
5: Synchronize device state and location with UAV;
6: **for** $i = 1, \cdots I$ **do**
7: Select chromosome matrix(Num);
8: **for** $j = 1, \cdots MatrixNum$ **do**
9: Recombination chromosome;
10: Mutation chromosome;
11: **end for**
12: Decode DNA;
13: Update Cost:
14:
15: $$SC = \sum_{n=1}^{n} \{\alpha \cdot T_n + (1-\alpha) \cdot E_n\}, n \in N;$$
16: Update Fitness;
17: Get Population(2Num) of Parent Population and the New Population;
18: **end for**
19: Invoke Algorithm 2 for extract task allocation strategy and location of FES;
20: Reach the destination with OSDK and start offloading on its tasks;
21: **end for**

Each chromosome corresponds to an individual. Encoding information consists of task allocation vector and UAV hover location. Each chromosome represents a potential solution (*i.e.,* task allocation strategy and UAV hover location) in the solution space. The population is the solution space that

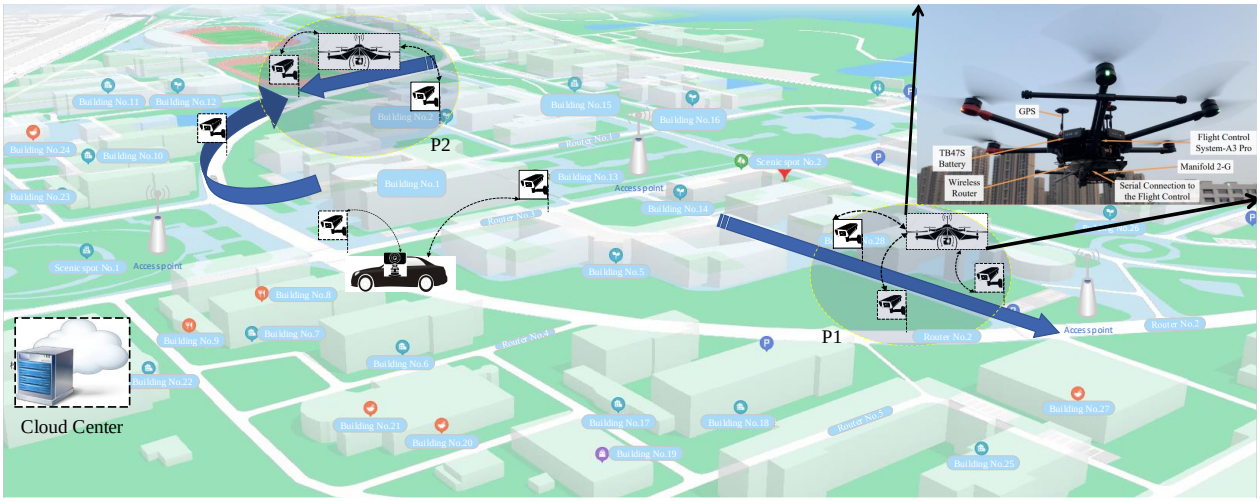


Fig. 5: An evaluation platform (Ampere) for FlexEdge. The blue arrow line represents the movement direction of detected objections. Symbols P1 and P2 represent the different UAV locations.

Algorithm 2 Extract task allocation strategy and FES location

Input: A list $[x_k, y_k, mark_1, mark_2, mark_3, mark_4, mark_5, mark_6, mark_7, mark_8]$: location of FES and task allocation strategy of each edge node.

Output: The location and the task allocation strategy is sent to the FES and each edge node, respectively.

Initialization: $sch = [mark_1, mark_2, mark_3, mark_4, mark_5, mark_6, mark_7, mark_8]$

1: Send location $[x, y]$ to the FES;
2: **for** $k = 1, \cdots K$ **do**
3: Send the allocation strategy $mark_k$ to edge node k;
4: Create the corresponding receive channels for transmitting data and logs;
5: **end for**

contains multiple sets of solutions. We can obtain the optimal solution by FGA to minimize the weighted sum of execution time and energy consumption (see Eqs. (14)∼(24)).

Let us suppose that K ground edge nodes are used in the platform. Each edge node requires a task allocation vector to guide the location of task allocation (local, cloud, and FES). Each individual in the optimized genetic algorithm is referred to as a solution vector, see Fig. 4. The first two data items are the coordinate of the UAV hover location and; each one in the rest of the items represents a task allocation strategy on each ground edge node.

The algorithm continues to execute until it meets the condition. When selecting the population matrix (Line 7), N individuals are recombined and mutated (Lines 9 and 10). The chromosome is then translated to allocation variables and location in Line 12. The objective function evaluates the current population, and then the optimal individuals and average fitness are recorded. Based on the fitness evaluation, Num individuals are independently selected from the current population. Num parent codes are crossed to obtain their offspring; then, Num cross individuals are mutated. The parent population and mutated cross-population are combined to obtain a population of Num. These Num individuals are selected from the combined population $2Num$ according to the fitness evaluation. Subsequently, we obtain a new population. Steps after initialization are loop performed until the evaluation function no longer raises or reaches the default iteration number. We use Algorithm 2 (decode operation in Line 17) to extract the task allocation strategy and location of FES. The system automatically synchronizes information at an edge node and the two-layer servers to execute the allocation strategy and control the UAV hover location. Algorithm 2 can provide a solution vector once. The vector includes UAV hover location and task allocation strategy for K edge nodes. The coordinate and task allocation strategy are sent to the UAV flight controller module and the corresponding edge node. Our work aims to design an efficient scheduling method to find an optimal task allocation strategy and UAV hover location minimizing the normalized weighted sum of execution time and energy consumption. The scheduling problem is usually transformed into a large-scale nonlinear combinatorial optimization problem and has been proven to be NP-hard, see Appendix A. The complexity of the problem presented in this study is also detailed in the Appendix B. In this study, we propose a flexible and lightweight genetic algorithm (FGA) based on a polysomy-strengthening elitist genetic algorithm in FlexEdge to minimize the weighted sum of the execution time and energy consumption. Our proposed method can adapt to the computation and energy-constrained scenarios in UAVs; it generates an entire task allocation policy, and completes the edge task processing in real-time. The reason that we choose a genetic algorithm to solve the problem is presented as follow.

Genetic algorithms (GAs) are among of the most popular evolutionary algorithms, being metaheuristic algorithms inspired by natural selection that belong to a larger class of evolutionary algorithms. GAs are very robust and can solve highly complex nonlinear problems (e.g., NP-hard problems) compared to traditional search algorithms such as dichotomous, Fibonacci, Newtonian, and parabolic methods. However, traditional random search and exhaustive search algorithms [64] are computationally intensive owing to the high complexity of the scheduling problem and exponential growth search space, making it a challenge to deploy in real-world platforms such as the FlexEdge. GAs have a relatively global solid search capability, especially when the crossover probability is relatively large, generating many new individuals and improving the global search range. These algorithms are primarily employed to generate high-quality optimization solutions and search problems by relying on biologically inspired operators such as mutation, crossover, and selection. GAs [63], [65], [66] are commonly used to find the optimal solutions to address a given computational problem that maximizes (or minimizes) a particular function. They are more efficient than random search and exhaustive search algorithms [64] while not requiring additional information about the problem. Although there is a lack of continuity, derivative, linearity, etc., in the condition, genetic algorithms can still find a solution to problem that some other optimization methods may not be able to handle. GAs are suitable for solving discrete problems, such as the task allocation strategy used in this study.

Our proposed FlexEdge is based on SEGA [67], which optimizes the strategies in the elite retention strategy, the mechanism of the selection operator, and the encoding of DNA, FGA has two advantages over canonical GAs.

- The selection operator employs a tournament selection strategy instead of the traditional roulette strategy. Many individuals are selected from the population at a time (put-back sampling), and the best one enters the offspring population. This operation is repeated until the offspring population size reaches that of the original population. The selection pressure for tournament selection can be adjusted according to the number of competitors. This method is efficient for code; and conduct on parallel architectures [68].
- The task allocation problem is a large-scale nonlinear combinatorial optimization problem in our study. The allocation of tasks is discrete, and our proposed algorithm encodes the individual DNA of the population into the form shown in Figure 4 to address the discrete problem.

In conclusion, these optimization approaches enable our proposed algorithm to converge globally, and the convergence

speed meets the requirements of battery limitations in UAVs. Experiments validate the advantages of our proposed method over other methods, as shown in Figs. 6, 7, 8.

In our proposed PSEGA-based algorithm, we encoded the first two data items as the coordinates of the UAV location. The last eight data items are the flag bits of the task allocation on the edge nodes (see Fig. 4). Our optimized genetic algorithm can obtain the approximate optimal task allocation strategy and UAV location in the edge environment within 1s while being over 98% as accurate as the exhaustive method. Our proposed method is applicable to the resource-constrained edge computing environments.

As one of the popular evolutionary algorithms, the genetic algorithm has a strong global search capability. When the crossover probability is large, it can generate many new individuals and improve the global search range [69]. The genetic algorithm has advantages in solving discrete problems such as the task allocation strategy in our work. The convergence of the genetic algorithm guarantees the optimal solution; meanwhile, its low complexity achieves real-time video analysis. FGA obtains the approximate optimal task allocation strategy and the UAV hover location within one second. The accuracy is higher than 98% of that in the exhaustive method.

VII. Experiment Setup

We set up a realistic prototype to compare our proposed FGA with alternative strategies.

A. Experimental Environment

In Fig. 5, we built a experimental platform, Ampere composed of edge nodes, an FES, and a cloud server. Table II lists characteristics of devices. This framework uses H.264 to encode the video's length, width, and frame rate in the system.

Local layer. Without loss of generality, we deployed eight edge nodes (*i.e.,* q_k) in Fig. 5. An edge node is composed of a camera and a development board that provides computing capability at edge nodes. We employ three Raspberry Pi 3 [70], two Raspberry Pi 4 B [71], one NanoPC-T4 [72], and two Jetson Nano [73]. Local video stream is used as the test data to ensure the same video task and data volume in each experiment. Each edge device (*i.e.,* q_k) connects to the FES via Wi-Fi while connecting to the cloud through Wi-Fi, 4G, and 5G networks, see Fig. 2.

FES layer. The UAV is DJI Matrice 600 pro, see Fig. 5. Its motion control programming uses DJI SDK 3.6.1. The flight control system is A3 pro. The NVIDIA TX2-based Manifold2-G [74] provides computing resources. The communication resource is Huawei 4G routing 2. The FES connects to the edge node through the Wi-Fi, see Fig. 2. The FES can periodically and dynamically move to the area where edge nodes have insufficient computing resources for video analysis. High computing latency and energy consumption are imposed on the local devices when all video is analyzed on the edge devices. Transferring video data to the cloud consumes bandwidth. A portion of tasks are allocated on the FES; subsequently, results are sent to the edge device in real-time, improving bandwidth and resources utilization in the system.

Cloud layer. The cloud server collects devices status for task allocation strategy and UAV hover location decision-making scheme. The server is featured with an Intel_Xeon(R) CPU E5-2630 v4 with 2.20GHz*40, 64-GB DRAM, and an NVIDIA Tesla P100. The cloud server receives allocated tasks and returns feedback to edge nodes (see Fig. 2). We conduct experiments in a 5G environment near a 5G cell tower, providing 4G network connection. The local campus wireless network is Wi-Fi. For each network condition, the available bandwidth is limited to the minimum bandwidth that each device uses in the test.

In Eq. (3), β_0, θ, R_0, P, and N_0 are set to -60 dB, 2, 100 MHz, 0.1 W, and –110 dBm, respectively. We use a power consumption monitoring module called iReader-opto's IM1253B, accurately measuring voltage, current, and power. This module connects to the development board using a UART interface. We employed a realistic transfer data volume and computing tasks in the platform. In a task allocation strategy, we normalize the execution time and energy consumption to guarantee the weight coefficient, which can avoid the impact of excessive energy consumption. Upload bandwidth 63 Mbps, 23 Mbps, and 5 Mbps are configured for 5G, 4G, and Wi-Fi networks, respectively. Extensive experiments were conducted to evaluate the performance of task allocation strategies in the three layers. Video analysis applications on ground edge nodes include face cascade classifier haar cascades (HAAR), DNNs in OpenCV, Dlib-based MMOD, and object detection YOLOv3. The first three applications are real-time face detection applications, where the computing resources increase as the same data volume at the local.

The last one is a high-performance object detection application. These applications are applicable for video analytics in real-world edge computing scenarios.

We study the execution time (see Table III and energy consumption (see Table IV) of each device with the tested video stream of 30 FPS and 720P resolution. Taking HAAR as an example, the size of a data stream is 3,600 frames in this test. Raspberry Pi 3 costs 530.0s and 3,136.4J. Raspberry Pi 4 costs 265.6s and 2,179.8J while NanoPC-T4 takes 260.1s and 1,548.9J. For Jetson Nano, the values are 258.1s and 1,243.0J. Manifold2-G has 150.8s and 3,373.2J. Besides, the cloud server costs 72.4s in this application.

In addition, detection applications deployed in three layers have the same recognition accuracy based on same model and parameters.

B. Alternative Task Allocation Strategies

We compare FGA with three basic task allocation strategies, *i.e.,* local computing, FES computing, and cloud computing, and six hybrid task allocation strategies, namely, the exhaustive method (EM), greedy algorithm based on simulated annealing (GA [75]), successive convex approximation (SCA [76]), alternating direction method of multipliers (ADMM [77]), differential evolution (DE [78]), PSO and deep reinforcement learning (DRL [79]).

For other evolutionary algorithms, particle swarm optimization (PSO) and differential evolution (DE) algorithms are

TABLE II: Devices and Characteristic

Device	CPU	GPU	Mem	OS	Num	Picture
Raspberry Pi 3 [70]	Broadcom BCM2837 4-Core @ 1.2GHz	*	1 GB	4.19.118-v7+	3	
Raspberry Pi 4B [71]	Broadcom BCM2711, 4-core @ 1.5GHz	*	4 GB	4.19.118-v7l+	2	
NanoPC-T4 [72]	2-Corex-A72@2.0GHz + 4-Corex-A53@1.5GHz	Mali-T864	4 GB	4.4.154	1	
Jetson Nano [73]	4-Corex-A57@1.43GHz	128 cuda core Maxwell	4 GB	4.9.140-tegra	2	
Manifold2-G [74]	2-Core 2.0 GHz Denver2 + 4-Core 2.0 GHz Cortex- A57	256 cuda core GHz NVIDIA Pascal 1.3	8 GB	4.4.38	1	
Server	Intel_ Xeon(R) CPU E5-2630 v4 @ 2.20GHz*40	3584 cuda core 875MHz NVIDIA Tesla P100*4	64 GB	4.4.0-139-generic	1	

* The device does not have this type of hardware.

TABLE III: Execution Time (second) of Video Analytics at Local

Time(s)	Pi 3	PC-T4	Pi 4	Nano	Manifold 2-G	Server
HAAR	530.0	260.1	265.6	258.1	150.8	72.4
DNN	4475.0	3168.1	1417.0	477.7	285.6	54.4
MMOD	20050.6	15750.4	9362.9	314.4	184.9	20.5
YOLO	3309.5	1811.9	2094.7	661.6	213.3	26.2

TABLE IV: Energy Consumption (Joule) of Video Analytics at Local

Energy(J)	Pi 3	PC-T4	Pi 4	Nano	Manifold 2-G
HAAR	3136.4	1548.9	2179.8	1243.0	3373.2
DNN	23693.4	13786.6	13354.5	2850.0	7317.1
MMOD	84479.4	81655.9	81566.3	1901.4	2142.2
YOLO	27059.2	13608.3	14822.7	3937.5	1385.3

popular for solving optimization problems. The PSO algorithm is suitable for path finding problems on graphs but incur excessive computational overhead for use in an edge environment. The emerging evolutionary algorithm DE has only two adjustable parameters that are not sufficiently flexible. It is challenging to determine the optimal solution under the accuracy requirement for a small population size. Our test results reveal that the differential evolution slowly obtains the optimum when the population is large.

Hybrid offloading strategies that use three-layer computing resources can allocate tasks on an edge node, the FES, or the cloud server.

It must be noted that there are three-type network connections (Wi-Fi, 4G, and 5G) used in the six hybrid task allocation strategies (EM, GA, SCA, ADMM, DE, DRL, and FGA). For example, notation 'net' in *GA-net*, net $\in$ {Wi-Fi, 4G, 5G} denotes that a network connection is used in the GA-based hybrid task allocation strategy. Without loss of generality, symbols *GA-4G, GA-5G*, and *GA-W* represent 4G, 5G, and Wi-Fi networks employed in the test for GA, respectively. The other five hybrid task allocation strategies also have the same definition under the three networks.

Local computing (*i.e.,* L) is used as the baseline. All video analysis tasks are executed on edge nodes without task offloading except uploading results.

FES computing. There are two types of task allocation strategies. First, all tasks are locally conducted on the FES, which is represented by *F*. Second, notation *F-W* means that all tasks are allocated to the FES from edge nodes via Wi-Fi.

Cloud computing. We use the symbol *C* to show that tasks are locally performed on the cloud. Notation *C-net*, net $\in$ {Wi-Fi, 4G, 5G} means that all tasks are allocated to the cloud layer from edge nodes through Wi-Fi, 4G, and 5G networks, respectively. Symbol 'net' in 'C-net' denotes the network connection. For example, *'C-4G'* denotes that a 4G network is used in this task allocation strategy.

EM is used to produce the best task allocation strategy and UAV hover location decision-making scheme in the solution, which validates the benefits of our proposed FGA.

GA employs a greedy algorithm based on simulated annealing, which accepts a solution worse than the current solution with a certain probability when iteratively updating a reliable solution. Afterward, this algorithm potentially jumps out of the local optimum.

SCA is an efficient algorithm that finds approximate solutions to optimization problems (*e.g.,* NP-hard problems) with provable guarantees on the distance of the returned solution to the optimal solution. The design and analysis of approximation algorithms crucially involve mathematical proof certifying the quality of the returned solutions in the worst case.

ADDM is a variant of the augmented Lagrangian scheme. This algorithm is a popular approach to addressing the linear coupling constraint. The linear constraint is added to the objective by using the augmented Lagrangian regularizer, and a gradient ascent step is used to update the dual variables in the dual problem. Rather than iterating until convergence, the algorithm immediately updates the dual variable and then repeats the process. Due to the limitation in its convex function, the algorithm replaces the UAV hover location decision-making scheme in our proposed FGA to obtain the task allocation strategy.

PSO (particle swarm optimization) [80], a computational

method, optimizes a problem, solving it using a population of candidate solutions called particles and moving them in the search space according to a mathematical formula based on their position and velocity. This method iteratively aims to improve a candidate solution with a given measure of quality. The local-best-known position of a particle determines its movement; the particle is also guided toward the best-known position in the search space. The best-known positions are updated as other particles find better positions, enabling the swarm to move toward the best solutions.

DE (differential evolution) [81] optimizes a problem by iteratively attempting to improve a candidate solution with regard to a given quality measure. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to their simple formulae. The candidate solution has the best score or fitness on the current optimization problem.

DRL combines reinforcement learning (RL) and deep learning. RL considers the problem of computational agent learning to make decisions by trial and error. Deep RL incorporates deep learning into the solution, allowing agents to make decisions from unstructured input data without manually engineering the state space.

Our proposed FGA. We proposed an enhanced elite retention-based multi-chromosome GA to obtain the task allocation strategy and UAV hover location.

In alternative task allocation strategies, the initial weights of energy consumption and execution time have the same values, *i.e.,* $\alpha = 0.5$.

C. Tested Metrics

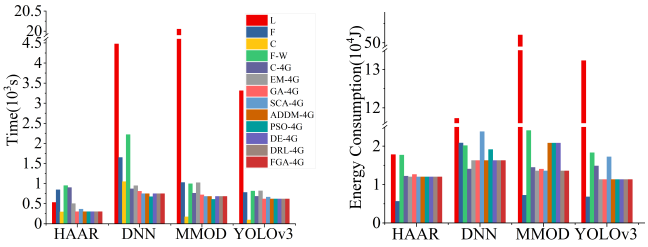
Our tested metrics include execution time, energy consumption, the number of tasks, and iteration times. First, we studied the execution time and energy consumption of single-layer allocation at the edge, FES, and cloud layers under hybrid task allocation strategies. Then, we observed the impact of networks (*e.g.,* Wi-Fi, 4G, and 5G) on the number of allocation tasks. Second, we tested the impact of different weights and networks on execution time, energy consumption, and task allocation. Third, we studied the influence of iterations (50, 100, 150, and 200) and networks on execution time and energy consumption.

VIII. Performance Evaluation

In FlexEdge, we conducted tests to study execution time, energy consumption, and the number of tasks under task allocation strategies (see Section VIII-A). Section VIII-B presents that the FGA performance varies with the weights of time and energy consumption. We studied the FGA iteration impacts on the performance in Section VIII-C.

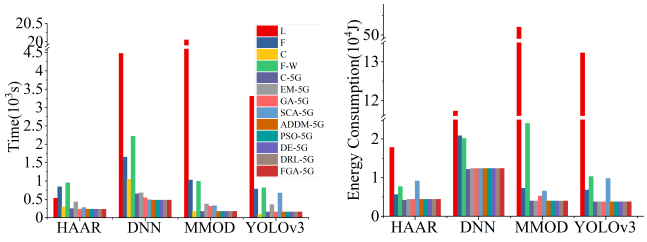
A. Impact of Task Allocation Strategies on Execution Time and Energy Consumption

We studied the performance of strategies L, F, and C under the same number of tasks. Symbols L, F, and C denote executing all video analytics tasks at the ground node, the FES, and the cloud server, respectively, without task offloading. We ignore energy consumption in the cloud because of its sufficient computing resources. The local computing mode (L) has at least one order of magnitude larger execution time and energy consumption than other methods in most cases. In addition, the FES has a lower computing capability than the cloud server.



(a) Execution time. (b) Energy consumption.

Fig. 6: The impact of allocation strategies on execution time and energy consumption under 4G.



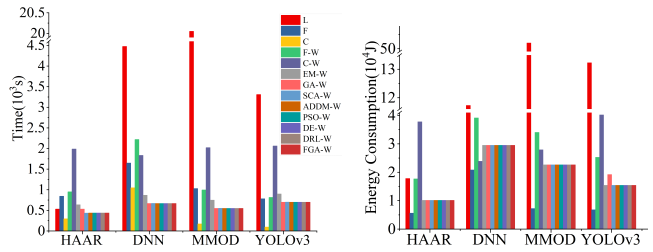
(a) Execution time. (b) Energy consumption.

Fig. 7: The impact of allocation strategies on execution time and energy consumption under 5G.

Fig. 6 shows the performance of four detection applications in terms of execution time and energy consumption under the 4G network. *FGA-4G* achieves the best performance. In FGA-4G, the weighted sum of the two metrics improves that of *C-4G* by up to 1.37×. Other hybrid allocation schemes have a higher performance than the cloud. This is because 4G network cannot meet the network speed requirements in the cloud. Even though many computing resources are in the cloud, the computing resources cannot be fully utilized because of the low transmission speed. In the hybrid offloading strategy, a portion of tasks is processed locally on ground edge nodes with available computing resources. The bandwidth for offloading tasks is allocated to edge nodes with less computing capability to transfer data to the FES or the cloud server.

Fig. 7 shows the performance of task allocation strategies in terms of execution time and energy consumption under the 5G network. *FGA-5G* achieves the best performance with a 1.19× improvement of the weighted sum of the two metrics compared with *C-5G*. Unlike the 4G network, the transmission latency of 5G is lower than the execution time on the cloud server. This means that the number of offloading tasks already exceeds the task processing capability of the cloud server. However, the computing resources of the ground edge nodes boost the performance of video analysis. Therefore, our proposed FGA also has better performance than cloud computing under the 5G network.

In Fig. 8, the Wi-Fi network between the edge node and



(a) Execution time. (b) Energy consumption.

Fig. 8: The impact of allocation strategies on execution time and energy consumption under Wi-Fi.

the cloud has a lower bandwidth (*i.e.,* 5 Mbps) than the 4G (*i.e.,* 23 Mbps) and 5G (*i.e.,* 63 Mbps) networks, leading to higher values of the cloud computing strategy (*C-W*) than other networks. *FGA-W* improves the performance of *C-W* by up to 3.37×. The Wi-Fi network has a significant impact on the HAAR compared to other networks. The reason is that HAAR is a lightweight task that requires less execution time and the transmission latency largely determines the total time.

In summary, hybrid allocation strategies provide better resource utilization than single-layer strategies. Results unveil that our proposed *FGA* is an efficient task allocation strategy; moreover, it can obtain an approximate optimal task allocation strategy and UAV hover location under the requirements of a real-time and resource-constrained environment.

We studied the performance of hybrid allocation strategies.

EM spends much time traversing all candidate task allocation strategies to find the optimal solution. The whole process takes approximately 210 seconds.

GA fails to obtain the optimal task allocation strategy and UAV hover location due to its local optimum nature, reducing the probability of finding the optimal task allocation strategy (58.3%) and increasing idle energy consumption. Compared with FGA, GA requires a similar time to obtain the task allocation strategy in most cases, but FGA can quickly determine the task allocation strategy and UAV hover location within one second. Besides, FGA avoids the local optimum solution problem in GA.

SCA finds a local optimal solution to the original problem by iteratively solving a series of convex optimization problems similar to the original problem. This method is extremely fast at obtaining the optimal solution, but it has similar accuracy to GA (50%). Alternative strategies in the test have a small time cost (within one second) due to the real-time requirements of video analytics. Except for EM (around 210 seconds), we can ignore the time cost to obtain the task allocation strategy and the UAV hover location compared to the task execution time (*e.g.,* at least 200 seconds) in the system.

ADMM is mostly used to address convex optimization problems with equation constraints. In this paper, we decompose the proposed problem into two subproblems, including the task allocation problem and UAV hover location problem. The task allocation problem, which is non-convex, cannot be addressed by ADMM. The UAV hover location problem can be converted into a convex optimization problem. Thus, we use a combination of genetic algorithms and ADMM to control the performance difference caused by the algorithms. Given the optimal strategy, the ADMM decision time is higher than one second due to its excessive iterations. FGA has higher accuracy in terms of UAV hover location than ADMM.

There are also some issues with the PSO algorithm. First, the particles tend to lose their local optima. Second, the PSO has a high likelihood of converging prematurely convergent to a local optima. Third, it trusts other particles, many of which can fall into the local optimum that other particles expect, thereby obtaining the local optima. Thus, the probability of finding the optimal task allocation strategy is reduced to approximately 83.4%. In summary, the PSO algorithm may quickly falls into local optima, making it difficult to obtain the global optimal solution. Even though PSO has a similar execution speed to our proposed algorithm, the accuracy of finding the optimal solution is low.

DE has too few configurable parameters to be flexible enough. Currently, fewer DEs use hybrid encoding methods. We find that the execution time of DEs is higher than one second when the number of populations is large, which affects the real-time performance of formulating the strategy. Otherwise, a small number of populations impairs the probability of obtaining the optimal task allocation strategy.

DRL not only spends minimal time obtaining the task allocation strategy and UAV location but also achieves an accuracy of around 99% to obtain the optimal strategy. FGA has similar accuracy to DRL. The drawbacks of DRL include the complexity of its simulation environment design and the high time cost for the cloud server to train the model. We cannot ensure the training time. It is difficult for this method to be flexibly applied in a mobile edge environment. FGA has advantages of low execution time (within one second) and high accuracy (98%). Even though there is no optimal solution, FGA can find an approximate optimal solution in a complex edge environment. Moreover, FGA can directly calculate both the task allocation strategy and the UAV location using system parameters. In contrast, DRL needs to retrain the model, and SCA must reprogram the formula. Finally, FGA quickly finds the approximate global optimization in resource-constrained realistic applications with low computing resources requirements.

In a word, video analytics tasks are preferably offloaded to the cloud server under a 5G network because the data transfer rate exceeds the task execution speed at the cloud. In Wi-Fi, video analytics tasks are easily allocated to the FES or executed locally. This is because the Wi-Fi bandwidth between the edge nodes and the cloud (*i.e.,* 5 Mbps) is much lower than that between edge nodes and the UAV (*i.e.,* 10 Mbps). Thus, most tasks are given priority to be allocated to the FES; then, some tasks are allocated to local devices. Finally, the remaining tasks are offloaded to the cloud server via the limited Wi-Fi network. Parallel EM can provide a superior task allocation strategy than FGA, but it requires more parallel computing resources than FGA. We find that FGA saves about 210s of idle time and energy consumption over EM to obtain the task allocation strategy under the same level of available computing resources.

Fig. 9 shows the number of tasks at the three layers under Wi-Fi, 4G, and 5G networks when execution time and energy

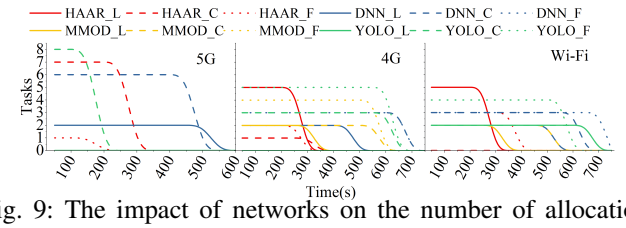
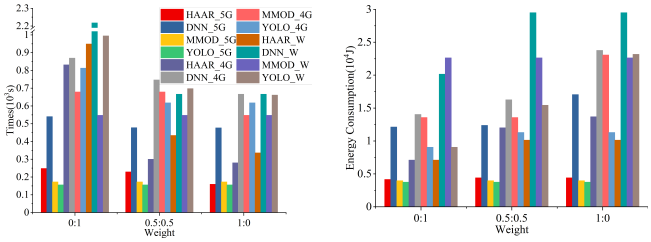


Fig. 9: The impact of networks on the number of allocation tasks. The solid line, dotted line, and dashed line represent local computing, mobile edge computing, and cloud computing, respectively.

consumption weight ratio is 1:1 in FGA. Results show the number of tasks at each layer during a cycle of task scheduling under the three networks. The task execution time belonging to the same scheduling strategy is similar, which indicates that FGA has efficient load balancing and resource allocation. Under the 5G network, most tasks are allocated to the cloud due to the high bandwidth. Tasks are allocated to the edge and the cloud under the 4G network because the bandwidth decreases. Under a Wi-Fi network, tasks are rarely allocated to the cloud. The low bandwidth enables transmission latency to increase sharply; thus, more tasks are allocated to the local and FES layers.

B. Impact of Weight on Execution Time, Energy Consumption, and Tasks



(a) Execution time. (b) Energy consumption.

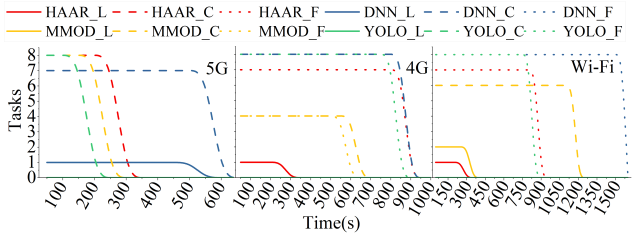
Fig. 10: Impact of weights on execution time and energy consumption in FGA.

Fig. 10 shows the execution time and energy consumption when we set different weights of the two metrics. As the weight of execution time increases, the execution time presents a steady downward trend in applications under the same number of tasks. Energy consumption presents an increasing trend. Thus, we can change the weight of execution time and energy consumption to meet different requirements of the two metrics in task allocation strategies for applications.

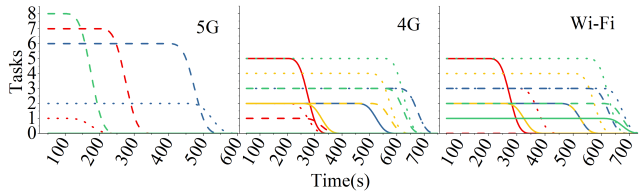
Fig. 11 presents the impact of weight on FGA in terms of execution time and energy consumption. To study the number of tasks, we configure the same scale on the y-axis. When the weight of execution time increases, the task execution time becomes shorter. Results show that the execution time with a weight of 1:0 is 19% and 20% lower than that of weights of 1:1 and 0:1, respectively.

C. Impact of Number of Iterations on Performance in FGA

Fig. 12 reveals the weighted sum of execution time and energy consumption under the three networks at each layer.



(a) Task allocation with a weight of 0:1.



(b) Task allocation with a weight of 1:0.

Fig. 11: Impact of weights on task allocation under the three networks.

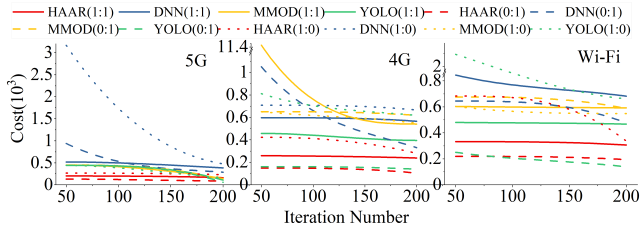


Fig. 12: Impact of different networks on the number of allocation tasks. A solid line, dotted line, and dashed line represent local computing, mobile edge computing, and cloud computing, respectively.

Because the evolution is featured with randomness, the execution time or energy consumption is not regular. Thus, we normalize the weighted sum of the two metrics. We study the cost of tasks when the weight ratio is 1:1. With the evolution algebra increment, the cost has an apparent decreasing trend, while the system achieves higher performance with an increase in the number of iterations.

Compared with other weight ratios, the weight ratio of the 1:1 based weighted sum of two metrics presents a stable downward trend under the 5G network, see Fig. 12. Results unveil that tasks are mostly allocated to local layer and FES layers as the network bandwidth decreases.

IX. Conclusions

Edge computing can optimize the execution time and energy consumption of video analytics in a mobile edge environment. We propose a flexible three-layer cloud-edge collaborative scheduling strategy based on a UAV named **FlexEdge** for computation-intensive and delay-sensitive video analytic in an area. We design execution time, energy consumption, and bandwidth models on the FES according to UAV mobility and resources at three layers. As an NP-hard problem, we proved the task allocation strategy based on execution time, energy consumption, and UAV hover location decision-making scheme. We propose a flexible and lightweight genetic algorithm (FGA) based on a polysomy-strengthening elitist

genetic algorithm in FlexEdge to address the problem. Then, we can achieve a task allocation strategy and UAV hover location decision-making scheme. The efficiency of FGA has 75.2% (on an average) improvement over the cloud computing model (the best performing single layer allocation strategy) in video analytics; meanwhile, FGA guarantees the load balancing among edge devices in the system. Compared with the exhaustive method, FGA improves the system performance by 22%, while the probability of finding the optimal task allocation strategy is 98%. Meanwhile, the performance of our proposed FGA is significantly outperforming other hybrid allocation strategies in many aspects. Thus, results validate that FlexEdge improves the performance and resources utilization for video analytics in an edge environment.

ACKNOWLEDGMENTS

This work was supported by the University Synergy Innovation Program of Anhui Province (GXXT-2019-007) and the National Natural Science Foundation of China (62072001, 61702004, 61572209).

REFERENCES

[1] H. Sun, W. Shi, X. Liang, and Y. Yu, "Vu: Edge computing-enabled video usefulness detection and its application in large-scale video surveillance systems," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 800–817, 2019.

[2] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 94–100, 2017.

[3] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5g networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[4] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proceedings of the Workshop on Mobile Edge Communications*, 2017, pp. 7–12.

[5] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[6] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[7] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2017.

[8] S. Ahmed, M. Z. Chowdhury, and Y. M. Jang, "Energy-efficient uav-to-user scheduling to maximize throughput in wireless networks," *IEEE Access*, vol. 8, pp. 21 215–21 225, 2020.

[9] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for wsns," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, 2015, pp. 1–6.

[10] T. Bai, J. Wang, Y. Ren, and L. Hanzo, "Energy-efficient computation offloading for secure uav-edge-computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6074–6087, 2019.

[11] J. Wang, K. Liu, and J. Pan, "Online uav-mounted edge server dispatching for mobile-to-mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1375–1386, 2019.

[12] D. Callegaro, S. Baidya, and M. Levorato, "A measurement study on edge computing for autonomous uavs," in *Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications*, 2019, pp. 29–35.

[13] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[14] J. Zhang, Y. Wu, G. Min, F. Hao, and L. Cui, "Balancing energy consumption and reputation gain of uav scheduling in edge computing," *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[15] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an uav network in mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[16] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the uav-enabled mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7808–7822, 2020.

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

[18] C. Yuan, Z. Liu, and Y. Zhang, "Uav-based forest fire detection and tracking using image processing techniques," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2015, pp. 639–643.

[19] S. Wan, J. Lu, P. Fan, and K. B. Letaief, "To smart city: Public safety network design for emergency," *IEEE access*, vol. 6, pp. 1451–1460, 2017.

[20] E. Cusumano, "Emptying the sea with a spoon? non-governmental providers of migrants search and rescue in the mediterranean," *Marine Policy*, vol. 75, pp. 91–98, 2017.

[21] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2016.

[22] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[23] C. Shao, S. Leng, Y. Zhang, A. Vinel, and M. Jonsson, "Performance analysis of connectivity probability and connectivity-aware mac protocol design for platoon-based vanets," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5596–5609, 2015.

[24] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 159–173.

[25] B. Liu, H. Huang, S. Guo, W. Chen, and Z. Zheng, "Joint computation offloading and routing optimization for uav-edge-cloud computing environments," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2018, pp. 1745–1752.

[26] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "Uavfog: A uav-based fog computing for internet of things," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017, pp. 1–8.

[27] A. Guillen-Perez, R. Sanchez-Iborra, M.-D. Cano, J. C. Sanchez-Aarnoutse, and J. Garcia-Haro, "Wifi networks on drones," in *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*. IEEE, 2016, pp. 1–8.

[28] J. Molina, D. Muelas, J. E. L. De Vergara, and J. J. García-Aranda, "Network quality-aware architecture for adaptive video streaming from drones," *IEEE Internet Computing*, vol. 24, no. 1, pp. 5–13, 2020.

[29] J. P. Macker and M. S. Corson, "Mobile ad hoc networking and the ietf," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 1, pp. 9–14, 1998.

[30] C.-K. Toh, *Wireless ATM and ad-hoc networks: Protocols and architectures*. Springer Science & Business Media, 2012.

[31] N. Kalatzis, M. Avgeris, D. Dechouniotis, K. Papadakis-Vlachopapadopoulos, I. Roussaki, and S. Papavassiliou, "Edge computing in iot ecosystems for uav-enabled early fire detection," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2018, pp. 106–114.

[32] Y. Y. Munaye, H.-P. Lin, A. B. Adege, and G. B. Tarekegn, "Uav positioning for throughput maximization using deep learning approaches," *Sensors*, vol. 19, no. 12, p. 2775, 2019.

[33] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-uav enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[34] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2018.

[35] S. ur Rahman, G.-H. Kim, Y.-Z. Cho, and A. Khan, "Positioning of uavs for throughput maximization in software-defined disaster area uav communication networks," *Journal of Communications and Networks*, vol. 20, no. 5, pp. 452–463, 2018.

[36] W. Ma and L. Mashayekhy, "Privacy-by-design task offloading for uav-mounted cloudlets," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 286–288.

[37] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "Uav-enabled mobile edge computing: Offloading optimization and trajectory design," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[38] Y. Liu, S. Xie, and Y. Zhang, "Cooperative offloading and resource management for uav-enabled mobile edge computing in power iot system," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 229–12 239, 2020.

[39] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji, "Computation offloading in beyond 5g networks: A distributed learning framework and applications," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 56–62, 2021.

[40] X. Chen, C. Wu, T. Chen, Z. Liu, H. Zhang, M. Bennis, H. Liu, and Y. Ji, "Information freshness-aware task offloading in air-ground integrated edge computing systems," *arXiv preprint arXiv:2007.10129*, 2020.

[41] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[42] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[43] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[44] F. Ono, H. Ochiai, and R. Miura, "A wireless relay network based on unmanned aircraft system with rate optimization," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7699–7708, 2016.

[45] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, 2020.

[46] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in uav-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[47] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[48] M. Hidayab, A. H. Ali, and K. B. A. Azmi, "Wifi signal propagation at 2.4 ghz," in *2009 Asia Pacific Microwave Conference*. IEEE, 2009, pp. 528–531.

[49] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in uav-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[50] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

[51] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[52] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[53] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 11–20.

[54] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for qos-aware web service composition," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 34–41.

[55] J. Huang, Y. Chen, C. Lin, and J. Chen, "Ranking web services with limited and noisy information," in *2014 IEEE International Conference on Web Services*. IEEE, 2014, pp. 638–645.

[56] Y. Haimes, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE transactions on systems, man, and cybernetics*, vol. 1, no. 3, pp. 296–297, 1971.

[57] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[59] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-uav-enabled load-balance mobile-edge computing for iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6898–6908, 2020.

[60] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected uav: Computation offloading and trajectory optimization," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[61] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in uav-enabled mobile edge computing networks with multiple service providers," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.

[62] Y. Qu, H. Dai, H. Wang, C. Dong, F. Wu, S. Guo, and Q. Wu, "Service provisioning for uav-enabled mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3287–3305, 2021.

[63] J. Carr, "An introduction to genetic algorithms," *Senior Project*, vol. 1, no. 40, p. 7, 2014.

[64] K. E. Kinnear, W. B. Langdon, L. Spector, P. J. Angeline, and U.-M. O'Reilly, *Advances in genetic programming*. MIT press, 1994, vol. 3.

[65] P. M. Vasant, *Meta-heuristics optimization algorithms in engineering, business, economics, and finance*. IGI Global, 2012.

[66] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.

[67] K. De Jong, "Genetic-algorithm-based learning," in *Machine learning*. Elsevier, 1990, pp. 611–638.

[68] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex systems*, vol. 3, no. 5, pp. 493–530, 1989.

[69] e. Jazzbin, "geatpy: The genetic and evolutionary algorithm toolbox with high performance in python," http://www.geatpy.com/, 2020.

[70] R. P. Foundation, "Raspberry pi 3 model b," https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, 2016.

[71] ——, "Raspberry pi 4 model b," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/, 2020.

[72] FriendlyArm, "Nanopc-t4," http://wiki.friendlyarm.com/wiki/index.php/NanoPC-T4, 2018.

[73] Nvidia, "Jetson nano," https://developer.nvidia.com/zh-cn/embedded/jetson-nano-developer-kit, 2019.

[74] DJI, "Manifold 2," https://www.dji.com/cn/manifold-2, 2019.

[75] P. E. Black, "Greedy algorithm," *Dictionary of Algorithms and Data Structures*, vol. 2, p. 62, 2005.

[76] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[77] J. Eckstein and D. P. Bertsekas, "On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1, pp. 293–318, 1992.

[78] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[79] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560*, 2018.

[80] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary computation*, vol. 25, no. 1, pp. 1–54, 2017.

[81] S. P. Lim and H. Haron, "Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions," in *2013 IEEE Conference on Open Systems (ICOS)*. IEEE, 2013, pp. 41–46.

[82] J. A. Dıaz and E. Fernández, "A tabu search heuristic for the generalized assignment problem," *European Journal of Operational Research*, vol. 132, no. 1, pp. 22–38, 2001.

[83] X. Dai, M. Li, and J. Kou, "The study on the time complexity of genetic algorithmsh," *Journal of systems engineering*, vol. 14, p. 1, 1999.

[84] K. S. Trivedi, *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons, 2008.

[85] J. R. Sampson, "Adaptation in natural and artificial systems (john h. holland)," 1976.

[86] D. Goddberg, "Genetic algorithms in search, optimization and machine learning," 1989.
[87] W. Feller, *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.
[88] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE transactions on neural networks*, vol. 5, no. 1, pp. 96–101, 1994.

APPENDIX A
NP-HARD PROBLEM PROOF

Our proposed strategy for the UAV-based task allocation includes two subproblems. The first one is the optimal task allocation strategy for computing tasks on different devices. This means we must decide whether a video analysis task at each device is executed locally or allocated to the FES (or the cloud). The second one is the UAV hover location. The optimal location of the UAV affects the task allocation strategy. For the second subproblem, when all tasks at the edge devices are allocated on an edge device or an edge server, the UAV's location is also determined. Meanwhile, there is no change in the UAV hover location in the current task execution cycle. Thus, we focus on the first subproblem in the decision results of a video task at an edge device. If we prove the first subproblem is NP-hard, the original problem must be NP-hard. In combinatorial optimization, we can reduce the optimal task allocation strategy to the generalized assignment problem (GAP) (Eqs. (14)) for the first subproblem. These N sub-tasks at N edge devices are pre-allocated to K devices ($N < K, K = N + 2$). The symbol C_{kn} represents the cost of task n-th allocated to device k^{th}. Our goal is to minimize the cost.

To make an analogy with the GAP, we define the task allocation indicator variable as

$$x_{kn} = \begin{cases} 1, & \text{the } k\text{-}th \text{ device is assigned to complete the } n\text{-}th \text{ task.} \\ 0, & \text{the } k\text{-}th \text{ device is not assigned to complete the } n\text{-}th \text{ task.} \end{cases} \tag{26}$$

Edge device q_k $(k = 1, 2, 3, \cdots, K)$ can execute one sub-task at most whereas the FES or cloud server can perform allocated sub-tasks, which must meet the following requirements

$$0 \leq \underset{1\leq k\leq N, 1\leq n\leq N}{\forall} x_{kn} \leq 1, 0 \leq \underset{N+1\leq k\leq K, 1\leq n\leq N}{\forall} x_{kn} \leq N; \tag{27}$$

Meanwhile, each sub-task *j* must be executed at the edge node, the FES, or the cloud server. Then, we have

$$\sum_{1\leq k\leq K, 1\leq n\leq N} x_{kn} = N \tag{28}$$

For the objective optimization problem in Eq. (14), the task must be completed within an analogical cycle. This means the longest completion time for all tasks is limited. For all tasks, we have

$$\max T_{kn} \leq T_{dl} \quad \forall k = 1, 2, \ldots, K, \forall n = 1, 2, \ldots, N \tag{29}$$

According to the execution time model, the execution time at the three layers should satisfy the constraints

$$\begin{aligned} &\max \left\{ T^L_{kn}, T^F_{(N+1)n}, T^C_{(N+2)n} \right\} \leq T_{dl} \\ &\forall k = 1, 2, \ldots, K-2, \forall n = 1, 2, \ldots, N \end{aligned} \tag{30}$$

Given a scheduling strategy, the total energy consumption of all sub-tasks can be expressed as

$$\begin{aligned} E_{total}(x) &= \sum_{k=1}^{K} \sum_{n=1}^{N} x_{kn} \cdot E_{kn} \\ &= \sum_{n=1}^{N} \Bigg(\sum_{k=1}^{K-2} x_{kn} \cdot E^L_{kn} + x_{(N+1)n} \cdot E^F_{(N+1)n} \\ &\quad + x_{(N+2)n} \cdot E^C_{(N+2)n} \Bigg) \end{aligned} \tag{31}$$

To make an analogy with the GAP, we simplify the optimization objective to minimize the energy consumption in Eqs. (27), (28), (29), and (30) constraints,

$$\begin{aligned} &\min E_{\text{total}}(x) \\ &\text{s.t. Eqs. (27), (28), (29), (30),} \end{aligned} \tag{32}$$

The first subproblem is to make an analogy with GAP in which n tasks are offloaded to m devices. Because energy consumption is the optimization objective, the problem is a nonlinear GAP (Eqs.(26)∼(32)), which can be reduced to an NP-hard problem according to prior work [82]. Thus, the problem of obtaining the optimal UAV hover location to minimize the normalized weighted sum of execution time and energy consumption is NP-hard. It is difficult for traditional methods to obtain the optimal solution for the large-scale nonlinear combinatorial optimization problem. The heuristic algorithm is one of methods to obtain the approximate optimal solution.

APPENDIX B
TIME COMPLEXITY ANALYSIS OF OUR PROPOSED FGA

We employed time complexity analysis of genetic algorithms based on the schema survival in prior work [83]. In the schema survival, a schema risk function and a stochastic reliability model are introduced to analyze the time complexity of genetic algorithms.

A. Schema hazard function

First, the schema survival is redefined. The survival of a schema *H* to the next generation means that at least *r* individuals of the schema *H* survive. Symbol *r* is a predefined small positive integer. Due to the random error of the genetic algorithm, the value of *r* is greater than one.

We define these unsatisfied chromosomes as bad patterns and assume that the population contains $m(t)$ individuals with bad patterns in generation t. Since the chromosome populations are initialized randomly, the proportion of bad patterns in the initialized population is determined first. Initialization is first performed T times. Each initialization calculates the fitness values of all chromosomes of the population in generation t by the objective function. Herein, the chromosome set is noted as $\mathrm{S}_N = \{1, 2, \cdots, N\}$. In our work, chromosome populations and objective functions are defined as

$$\begin{aligned} &\{V_n = [x, y, \alpha_1, \alpha_2, \cdots, \alpha_{\mathrm{M}}], n \in \mathrm{S_N}\} \\ &sc_{v_m} = \sum_{x,y;n=1}^{\mathrm{M}} \{\alpha_m \cdot T_m + (1 - \alpha_m) \cdot E_m\} \end{aligned} \tag{33}$$

The fitness of the i-th individual V_i in the population is $f_i = sc_{v_i}$, and the set of fitnesses of all chromosomes is $\{f_i, i \in$

$\mathrm{S}_N\}$. The fitness sequence is rearranged in ascending order of fitness values and is denoted as $S = \{f_j, j \in \mathrm{S}_N\}$. The last three constraints of the system model are used to partition the sequence S. A chromosome is a bad pattern if the generated chromosome cannot satisfy the requirements of each device in terms of maximum delay and energy. After each initialization, we can obtain the number of bad patterns which is assumed to be D_j for the jth initialization.

Thus, after t initializations, the proportion of bad patterns to the initialized population is denoted as

$$p_b = \sum_{j=1}^{t} D_j/(t \cdot N) \tag{34}$$

The number of bad patterns in the initial population $m(0) = P_b \cdot N$, when the genetic algorithm ends, and the number of bad patterns tends to zero.

The selection operator used in this paper is the tournament operator, which is a type of put-back sampling. A k-competition is a one-time sampling of k individuals from the total population. The best of these individuals is selected and placed in the set of the next generation population. This operation is repeated several times for as many individuals as needed. Assume that the population of generation t has N chromosomal individuals and the total number of individuals in bad mode is m(t). Under the bid-race operator, to select a bad pattern in one experiment, all K chromosomes must be selected as bad patterns. The scheme of selecting k chromosomes at n chromosomes has a kind of $\begin{pmatrix} N \\ K \end{pmatrix}$. The number of scenarios in which K chromosomes are selected among m(t) bad patterns is $\begin{pmatrix} m(t) \\ K \end{pmatrix}$. Thus, in generation t, the probability of the bad model being selected in one experiment is represented as

$$P_s = \begin{pmatrix} m(t) \\ K \end{pmatrix} / \begin{pmatrix} N \\ K \end{pmatrix} \tag{35}$$

In this paper, we use a binary tournament while the number of chromosomes N of the chromosome population is set to a larger value and K is small with respect to N. Then, from the above factors, we approximate the selection probability as

$$P_s = (\mathrm{m(t)}/N)^2 \tag{36}$$

The population N is divided into two disjoint sets, *i.e.,* the set R consisting of H bad-mode individuals and the set $R^{'}$ consisting of N-H individuals of superior mode. The selection of the whole population is considered as the result of N mutually independent Bernoulli experiments. In each experiment, the probability that an individual in any set R is selected is P_s. In the superior mode, the probability to select an individual is $1 - \mathrm{P}_s$. In the bad mode, the probability to select an individual in N experiments is

$$\begin{pmatrix} N \\ x \end{pmatrix} p_{\mathrm{s}}^{x} \left(1 - p_{\mathrm{s}}\right)^{N-x}$$

where x is the number of individuals from the set. In the genetic algorithm, some individuals are destroyed by the crossover or the mutation operators, whereas others continue to survive. The probability of each surviving individual is

$$p_{\mathrm{surv}} = 1 - p_c \cdot \frac{W(H)}{L-1} - o(H) \cdot p_m \tag{37}$$

where p_c is the crossover probability, p_m is the mutation probability, L is the chromosome length, $W(H)$ is the defined length of schema *H*, and $o(H)$ is the factorial of schema *H*.

When the number of surviving individuals follows a binomial distribution, the survival probability of individual j (the set of x selected individuals) in crossover and variation operators is denoted as

$$\begin{pmatrix} x \\ j \end{pmatrix} p_{\mathrm{surv}}^{j} \ \left(1 - p_{\mathrm{surv}}\right)^{x-j}$$

Thus, the probability of more than *r* surviving individuals in schema *H* is

$$\sum_{j=r}^{N} \sum_{x=j}^{N} \begin{pmatrix} N \\ x \end{pmatrix} p_s^x \left(1 - p_s\right)^{N-x} \begin{pmatrix} x \\ j \end{pmatrix} p_{\mathrm{surv}}^{j} \ \left(1 - p_{\mathrm{surv}}\right)^{x-j}$$

Then, the probability of no more than $(r-1)$ individuals who cannot survive is

$$1 - \sum_{j=}^{N} \sum_{x=j}^{N} \begin{pmatrix} N \\ x \end{pmatrix} p_s^x \left(1 - p_s\right)^{N-x} \begin{pmatrix} x \\ j \end{pmatrix} p_{\mathrm{surv}}^{j} \ \left(1 - p_{\mathrm{surv}}\right)^{x-j}$$

The above equation is a conditional probability that schema *H* survives in generation t and perishes in generation $t+1$. Let the probability be $Z(t)\Delta$, where $Z(t)$ is defined as the risk rate [84] in reliability theory. Δt is the time interval; thus, $Z(t)\Delta t$ denotes the conditional probability that schema *H* survives to time *t* and is destroyed in the time interval Δt. Obviously, the time interval in the genetic algorithm means the generation interval; thus, there is Δt=1 and

$$\mathrm{Z(t)} = 1 - \sum_{j=}^{N} \sum_{x=j}^{N} \begin{pmatrix} N \\ x \end{pmatrix} p_s^x \left(1 - p_s\right)^{N-x} \begin{pmatrix} x \\ j \end{pmatrix} p_{\mathrm{surv}}^{j} (1 - p_{\mathrm{surv}})^{x-j} \tag{38}$$

It must be noted that there is $Z(t) \in [0, 1]$; at the start of the genetic algorithm, $(t = 0), Z(t)$ is a smaller positive value.

According to Holland schema [85], [86], there is

$$m_H(t+1) = m_H(t) \cdot \left(\frac{f_H}{f}\right) \cdot p_{\mathrm{surv}} \tag{39}$$

Herein, f_H is the average fitness of schema *H* at generation t and $\overline{f}$ is the average fitness of the population at generation t. We assume that there is $f_H = (1+c)\overline{f}$, and c is a constant; then, $m_H(t+1) = m_H(t) \cdot (1+c) \cdot p_{surv}$. By way of recurrence, we can obtain

$$m_H(t+1) = m_H(0) \cdot (1+c)^{t+1} \cdot p_{\mathrm{surv}}^{\mathrm{i}+1} \tag{40}$$

According to Eqs. (34) and (40), For a schema that has a smaller mean fitness than the mean fitness of the population $(c < 0)$, when there is $t \to \infty, p_s \to 0, m_H(t) \to 0$; then, we have $Z(\propto) \to 1$. Otherwise, when there is $t \to \infty, m_H(t) \to N, p_s \to 1$; and we have $Z(\propto) \to 1 - \sum_{j=r}^{N} \begin{pmatrix} N \\ j \end{pmatrix} p_{\mathrm{surv}}^{j} (1 - p_{surv})^{N-j}$.

Based on Eqs. (34), (36), (39), and (40), we have

$$p_s = k_1 \cdot k^t \tag{41}$$

where $k_1 = (P_b)^2, k = ((1+c)\cdot p_{\text{surv}})^2$

Combining Eq. (41) and Eq. (38), we can obtain

$$Z(t) = 1 - \sum_{j=r}^{N}\sum_{x=j}^{N} \left(\begin{array}{c} N \\ x \end{array} \right) \left(k_1 k'\right)^x \cdot \left(1 - k_1 k^t\right)^{N\cdot x} \cdot \left(\begin{array}{c} x \\ j \end{array} \right) p_{\text{surv}}^j \left(1 - p_{\text{surv}}\right)^{x-j} \tag{42}$$

which is also rewritten as

$$\frac{N}{(N-x)!j!(x-j)!} \cdot \left(1 - k_1 k^t\right)^{N-x} \cdot \left(k_1 k^t p_{\text{surv}}\right)^j \cdot \left(k_1 k^t - k_1 k^t p_{\text{surv}}\right)^{x-j}$$

If we have large *N*, $1 - k_1 k^t$ and $k_1 k^t p_{\text{surv}}$ becomes small. When symbols $N(1 - k_1 k^t)$ and $N k_1 k^t p_{\text{surv}}$ are moderate, the above polynomial distribution can be approximated by a multivariate Poisson distribution [87]. Then, we have

$$\begin{array}{c} e^{-N\left(1\cdot k_1 k^t + k_1 k^t p_{\text{surv}}\right)} \cdot \left[N\left(1 - k_1 k^t\right)\right]^{N-x} \cdot \left[N k_1 k^t p_{\text{surv}}\right]^j \cdot \left((N-x)!j!^{-1}\right) \\ = p(t). \end{array} \tag{43}$$

Obviously, the risk function can be approximated by the following equation

$$Z(t) = 1 - e^{-\lambda(t+h)} \tag{44}$$

where λ, h are constants that depend on N, p_c, p_m, r and schema *H*.

We can obtain λ and h as follows

$$\frac{\partial E}{\partial \lambda} = 0, \frac{\partial E}{\partial h} = 0 \tag{45}$$

in which there is $E = \sum_i \left\{Z_1(t_i) - Z_2(t_i)\right\}^2, Z_1(t_i) = 1 - \sum_{j=r}^{N}\sum_{x=j}^{N} p(t_i), Z_2(t_i) = 1 - e^{-\lambda(t_i+h)}$. Thus, we obtain that the approximation increases as generation *t* increases.

According to the definition of the risk function, let the reliability $R(t)$ of schema *H* be the probability of the model survival to generation *t*. By mathematical induction, we can obtain

$$R(t+1) = R(0)e^{-\lambda(t(t-1)/2 + t^h)} \tag{46}$$

It is assumed that the boundary condition is $R(0) = 1$, meaning that the schema cannot perish in the genetic algorithm. Then, we have

$$R(t+1) = e^{-\lambda(t(t-1)/2 + t^h)} \tag{47}$$

The analysis mentioned above verifies that we have $R(\propto) = 0$.

B. Time Complexity

Let $M = P_b \cdot N$ be the number of individuals of the bad schema at *t=0*. These *M* individuals survive with the probability of *R(t)*. Otherwise, these individuals perish with the probability of *1-R(t)*.

If the random variable *Q(t)* represents the surviving number of individuals of the schema at generation *t*, *Q(t)* meets the binomial distribution

$$\begin{array}{c} \text{Prob}(Q(t) = q) = (M!/(q!(M-q)!) \cdot [R(t)]^q [1 - R(t)]^{M-q} \\ q = 1, 2, \cdots, M. \end{array} \tag{48}$$

The expected number of bad schemes surviving at generation t is given by the mean of the binomial distribution, *i.e.,* $MR(t)$. When the genetic algorithm iterates indefinitely, the bad schemes can perish. Time *T* for the bad schemes to perish can be calculated when we have $MR(t) = 0$.

Let y_i and y_f denote the expected number of schemes in the population at the beginning and the end of the genetic algorithm, respectively. We assume that symbol *t* is the iteration period of the genetic algorithm when a certain degree of convergence is reached. Then, we have

$$yR(T) = y_f \tag{49}$$

Combining Eq. (47) with above equations, we have

$$t = \frac{(2h-1) + \sqrt{(1-2h)^2 - 8(h-1) + \frac{8}{\lambda}\ln\frac{y_f}{y_i}}}{2} \tag{50}$$

Each iteration of the genetic algorithm is divided into the following four parts, calculating individual fitness, selection, crossover, and variation. Let us assume that the number of chromosomes is N and the chromosome length is M. The overhead of calculating chromosome fitness is related to the chromosome length M. In this paper, the total fitness of the chromosome population is calculated as M×N. In the selection phase, we use a binary tournament selection strategy. The time overhead of this selection algorithm is related to the number of chromosomes in the population as 3N. For the parent selection in the previous step, partitioning is performed; and then crossover is executed to obtain new progeny individuals. The time for partitioning and pairing chromosome sets is N. Crossover is the exchange of some genes according to some crossover pattern. We use a two-point crossover operator, so the time overhead of crossover is 2N. Finally, we use a mutation operator, which randomly mutates one chromosome gene. The time overhead of mutation of the whole chromosome population is N. Therefore, the time overhead of one iteration is $(7 + M) \cdot N$.

Then, time complexity of the algorithm is represented as

$$T = (7 + M) \cdot N \cdot t \tag{51}$$

We can use the value of T to calculate the time complexity of a genetic algorithm or the average rate of schema processing.

Our proposed FGA is based on the polysomy-strengthening elitist genetic algorithm; then, we can obtain the time complexity of FGA according to the above description.

Based on the schema survival, we apply the schema risk function into the complexity analysis of the genetic algorithm. Then, we establish a stochastic reliability model. According to our objective model, we first defined the criterion for the bad schema in our work. Then, we count the proportion of bad schemas in the randomly initialized population. The initialization proportion of bad schema and the tournament selection operator are combined to design the risk function of bad mode in the FGA. Using the risk function, we can obtain the probability of a schema survival to each generation, in which we introduce the schema reliability, and derive the reliability function of the bad schema in this paper. Finally, the evolutionary time complexity of the FGA can be calculated according to the initialization ratio of the bad schema and its reliability function. In this paper, the extinction of bad schemas leads the evolutionary cycle to finish. When having the time

complexity of the evolutionary cycle, we analyze the intra-population computational time complexity of one evolutionary process. We can obtain the time complexity of the proposed FGA by combining the above two terms in this paper.

APPENDIX C
CONVERGENCE OF THE PROPOSED FGA

The global convergence of the genetic algorithm is defined as follows.

Definition 1: Let $Z_i = \max\left\{f\left(\Pi_K^{(n}(i)\right) : K = 1, 2, ..., n\right\}$ be a sequence of random variables that represent the best fitness in the state of time step t. The genetic algorithm converges to the global optimal solution if and only when $\lim P\ \{Z_t = f^\star\} = 1$, where $f^\star = \max\{f(b) \mid b \in IB^l\}$.

The canonical genetic algorithm (CGA) refers to a genetic algorithm that uses binary coding, proportional selection operators, single-point crossover operators, and simple mutation operators used in our proposed FGA. Based on prior work [88], we can conclude the global convergence of CGAs as follows.

Theorem C.1: CGA cannot converge to the global optimal solution.

Lemma C.1: The genetic algorithm that retains the best individual before selection eventually converges to the global optimal solution.

Lemma C.2: The genetic algorithm that retains the best individuals after selection eventually converges to the global optimal solution.

Combining Lemma C.1 and Lemma C.2, we can achieve the following theorem.

Theorem C.2: The genetic algorithm with optimal retention operation must converge to the global optimal solution.

In this paper, the genetic algorithm uses an elite retention operator during the evolutionary process according to the mechanism of elite retention. The optimal individual DNA that has emerged so far is not lost and destroyed by selection, crossover, and mutation operations. Our task allocation strategy and UAV location are encoded into the DNA of the population individuals. The optimal offloading strategy and UAV location obtained from each iteration are retained until the cycle is stopped when the objective function is no longer rising or the default number of iterations is reached. Thus, we can obtain the optimal offloading strategy and UAV location, which is convergent according to Theorem 2.

The proofs of the above Lemma and Theorem can be found in prior work [88]. Our proposed FGA is based on the polysomy-strengthening elitist genetic algorithm. According to the proof of the convergence of GA, we find that our proposed FGA is convergent.

In conclusion, canonical genetic algorithm cannot converge to the global optimal solution. But the genetic algorithm with optimal retention operation must converge to the global optimal solution. In this paper, the genetic algorithm uses the elite retention operator in the evolutionary process according to the mechanism of elite retention. Then, we can find that our proposed FGA is convergent.

Hui Sun received his Ph.D. degree from Huazhong University Science and Technology in 2014. He is an associate professor in computer science at Anhui University. His research interests include computer systems, edge computing, and non-volatile memory-based storage systems.

Bo Zhang, born in 1997. He is currently pursuing an M.S. degree in Anhui University. His main research interests include edge computing, edge intelligent, and storage systems.

Xiuye Zhang, born in 1997. He is currently pursuing the M.S. degree in Anhui University. His main research interests include edge computing and storage systems.

Ying Yu, born in 1996. She is currently pursuing an M.S. degree in Anhui University. Her main research interests include edge computing.

Kewei Sha is an Associate Professor of Computer Science and Associate Director of Cyber Security Institute at University of Houston-Clear Lake (UHCL). He received Ph.D. in Computer Science from Wayne State University in 2008. His research interests include Internet of Things, Cyber-Physical Systems, Edge Computing, Security and Privacy, and Data Management and Analytics. His research has been supported by NSF, NASA, UHCL and OCU. Dr. Sha has served as an Associate Editor of IEEE IoT Journal, Elsevier Smart Health and Springer Computing, a Guest Editor at several prestigious international journals, and an organizing committee member of many conferences. He is a senior member of both ACM and IEEE.

Weisong Shi received the BS degree from Xidian University in 1995, and the Ph.D. degree from the Chinese Academy of Sciences, in 2000, both in computer engineering. He is a Charles H. Gershenson distinguished faculty fellow and a professor of computer science with Wayne State University. His research interests include edge computing, computer systems, energy-efficiency, and wireless health. He is a recipient of the National Outstanding PhD dissertation award of China and the NSF CAREER award. He is a fellow of the IEEE and ACM distinguished scientist.