Delay-Aware Cooperative Task Offloading for Multi-UAV Enabled Edge-Cloud Computing

Zhuoyi Bai, Yifan Lin, Yang Cao, *Member, IEEE*, and Wei Wang, *Senior Member, IEEE*

Abstract—Unmanned aerial vehicle (UAV) has received tremendous attention in the area of edge computing due to its flexible deployment and wide coverage accessibility. In weak infrastructure scenarios, multiple UAVs can form on-site edge computing clusters to handle the real-time tasks. Further, a multi-UAV enabled edge-cloud computing system is coined by cooperating the UAVs with remote cloud, which provides superior computing capability. However, the uneven distribution of tasks makes it difficult to meet the real-time requirements when load balancing is unavailable. To address above issue, a delay minimization problem for multi-UAV enabled edge-cloud cooperative offloading is investigated in this paper. The problem is formulated as a non-convex problem based on models that reflect characteristics of the system, such as ubiquitous network congestion, air-to-ground wireless channel and cooperative parallel computing. An efficient cooperative offloading algorithm is proposed to address the problem. Specifically, convex approximation is applied to make the original problem tractable, and Lyapunov optimization is utilized to make online task offloading decisions. Finally, the correctness of the models are verified through a practical UAV-edge computing platform. Simulations based on measurement results and real-world datasets indicate that, the proposed algorithm fully utilizes the available energy to significantly reduce the tasks' completion delay.

Index Terms—Task offloading, unmanned aerial vehicle-enabled edge computing, delay optimization

1 Introduction

Unmanned aerial vehicle (UAV) has been widely used in special scenarios by telecommunication companies such as Qualcomm, Ericsson and China Mobile [1]. When performing monitoring and rescuing missions in the post-natural disaster scenarios (e.g., fixed infrastructures have been disrupted by earthquakes), UAV is able to fly to the site to collect essential data for subsequent risk assessment. The missions demand for the real-time analysis of the complicated data. Therefore, the concept of UAV-EC (i.e., UAV enabled edge computing) has been proposed [2]. The main feature of edge computing (EC) is to push computing to the network edges (e.g., base stations and access points) so as to enable computation-intensive and latency-critical applications at the resource-limited scenarios. Compared to cloud computing, UAV-enabled edge computing can effectively reduce the transmission delay caused by extremely long distance [3]. As Huawei's whitepaper shows, UAV technology combined with edge computing will enable emerging areas like AI and remote sensing to new levels of automation and new types of analytic solutions [4].

Extending single node UAV-EC to multiple UAVs is an effective method to strengthen the available computation capability [5]. Moreover, a multi-UAV enabled EC system can be further augmented by connecting to the remote cloud, which forms a multi-UAV enabled edge-cloud computing system. The system does not violate the post-natural disaster scenarios assumption. Multi-hop UAV relay network enables the UAVs to communicate with the remote Base Station (BS) [6]. Besides, the combination of UAV-EC and cloud computing is to realize complementary advantages. Edge computing has extremely low transmission delay, and cloud computing has abundant computation capability. When the load of tasks exceeds the computation capability of the UAV cluster, the exceeded tasks will be offloaded to the ground BS through air-to-ground link. Then the BS forwards the tasks to the remote cloud through the wired link.

Nevertheless, computation capabilities and task loads among UAVs are commonly following uneven distribution [7]. The unbalanced loads make part of UAVs endure a large amount of tasks with others in idle. Then the most task-heavy edge node leads to an extremely long computation delay. As a consequence, the real-time requirements may not be satisfied. In order to reduce the system delay, several theoretical researches have been proposed formerly from different perspectives such as trajectory optimization, resource allocation, etc [8], [9]. However, the methods are unable to adjust the computation loads among UAVs fundamentally. Thereby, in this paper, we investigate the task cooperative offloading strategy to balance the loads. Cooperative task offloading is not well investigated in the scenario of multi-UAV enabled edge-cloud system, but it has been proved as a critical factor for the performance improvement in EC systems [10]. UAVs

• *The authors are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: {zhuoyi_bai, yifan_lin, ycao, weiwangw}@hust.edu.cn.*

Manuscript received 1 June 2022; revised 10 October 2022; accepted 16 December 2022. Date of publication 27 December 2022; date of current version 8 January 2024.
This work was supported in part by the National Key R&D Program of China under Grant 2019YFB1803400 and in part by the National Natural Science Foundation of China (NSFC) under Grants 62271224, 62071194, 61925105, 62071192, and 61871178.
(Corresponding author: Yang Cao.)
This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2022.3232375, provided by the authors.
Digital Object Identifier no. 10.1109/TMC.2022.3232375

with less computation capability can offload part of workload to other UAVs to improve the overall efficiency of the system. Though cooperative offloading algorithms has been studied in the traditional ground edge-cloud systems [11]. the algorithms are not compatible with the multi-UAV enabled edge-cloud systems. The two kinds of systems have different characteristic in terms of communication, computing and power supply.

Compared to traditional edge-cloud computing at the ground, introducing UAVs brings new challenges: (i) Network congestion is ubiquitous in UAV wireless networks, especially when the UAV is streaming data [12]. (ii) Air-to-ground wireless backhaul is different from the wireless communications at ground. Accurate air-to-ground channel should be adopted to model the communication between UAV and the BS. (iii) Unlike the ground scenarios with massive tasks, multiple UAVs usually work together to handle a single mission. New model of mission completion delay should be carefully designed when the cooperation among UAVs are available. (iv) The difficulty of recharging UAVs' batteries during a mission results in more strict energy constraints.

Besides, existing researches which consider the UAV-enabled edge computing also have some aspects that are not covered. Multi-UAVs edge computing neglects the cooperation with the remote cloud [5], [13], [14], [15], resulting in an extremely different model from multi-UAV enabled edge-cloud computing systems. Meanwhile, although some researches consider the UAV enabled edge-cloud systems, most of them are with the objective to minimize the energy consumption [16], [17]. Few researches focus on the delay optimization in UAV enabled edge-cloud computing [18], but the essential factors (e.g., topology, network congestion, resource reservation based on virtualization, and the stochastic occurrence of practical tasks) have not been considered. As a consequence, an efficient cooperative offloading scheme in multi-UAV enabled edge-cloud computing should take into account more aspects than in existing researched scenarios.

In this paper, we comprehensively consider the characteristics of the UAV-EC systems, and propose a cooperative framework to solve the problem of uneven tasks distribution. Meanwhile, according to our knowledge, this is the first time to utilize online tasks offloading to minimize the total service delay in the multi-UAV enabled edge cloud computing framework. The main contributions of this paper are summarized as follows:

- A more comprehensive and realistic model focusing on multi-UAV enabled edge-cloud tasks offloading is proposed. Essential aspects corresponding to the practical systems have been introduced, to avoid the deviation from reality.
- Online optimization is adopted for the first time to optimize the service delay of multi-UAV enabled edge-cloud computing systems. An algorithm based on Lyapunov optimization and convex approximation is proposed to obtain the delay-optimal task offloading decision, with rigorous mathematical proofs.
- A real-world UAV enabled edge computing platform is constructed to guide the model verification. Measurements based on the platform indicate the correctness of the proposed model. Furthermore, utilizing the real-world datasets, data-driven simulation experiments have been performed to corroborate that the proposed algorithm can reach a near-optimal performance on system delay.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the system model, then Section 4 formulates the delay-aware cooperative offloading problem and proposes the Lyapunov optimal-based cooperative algorithm. Performance evaluation has been implemented in Section 5. Finally, conclusions are provided in Section 6.

2 Related Work

UAV-enabled edge computing attracts the wide attention of industry and academia. DJI, as the largest consumer rotary-wing UAV company in the world, has launched the Manifold 2 high-performance onboard computer[1] in 2019, to perform complex computing tasks and advanced image processing literally on the fly. Qualcomm produced the Flight RB5 5 G platform[2] as the world's first 5 G and AI-enabled UAV platform for accurate edge inferencing that reduces processing time for mission-critical UAV applications. Meanwhile, Verizon's researchers are testing how edge computing and artificial intelligence can help UAVs detect, interpret and respond to changing weather conditions in real time [19]. In addition, multiple UAV-enabled edge nodes can be utilized to explore larger areas in the scenarios.

Several schemes are also proposed to improve the UAV-enabled edge computing system's performance. Zhou et al., focused on the secrecy capacity problem of the UAV-enabled EC systems to explore the trade-off between the security and latency [8]. Trajectory optimization of the UAV edge nodes has been considered in [9], [20], [21] to jointly design the computation and communication strategies. Similarly, UAV's placement problem in the UAV-aided EC networks was investigated in [22]. Different from these work, this paper attempts to achieve performance improvement from the perspective of task offloading.

Zhou et al., studied the offloading algorithm of single UAV-enabled EC system to achieve computation rate maximization [2]. However, computation capacity of the edge computing units carried by single UAV may be insufficient to meet the requirement of large amount of computation tasks. Extending single UAV to multiple UAVs is an effective solution to reinforce the computation capacity. Specifically, Ma et al., modeled the multi-UAV EC system and proposed a potential game-based cooperative offloading algorithm, in which UAVs can offload the tasks to each other, to achieve the optimal computation delay [5]. However, game-based cooperative offloading ignores that, extending the EC system causes the parallel computing, i.e., the tasks completion delay depends on slowest edge node among multiple UAVs. The ignorance leads to a disconnect between the algorithm and the actual scenario. Apart from multi-UAV enabled edge nodes, another method was proposed to expand the computation capacity

1. https://www.dji.com/manifold-2
2. https://www.qualcomm.com/products/flight-rb5-platform

by incorporating cloud server. The wide coverage of UAV's communication module enables UAV to offload part of task workload to the remote cloud for alleviating the computation pressure [23].

As a consequence, Luo et al., requested the multiple UAVs for offloading all the computation tasks to the cloud server [24]. However, the long-distance and multi-hop communication brings a large overhead. For this consideration, Jeong et al., introduced an offloading strategy that single UAV can not only offload tasks to the remote cloud but also compute part of the tasks locally [21]. Combining the multi-UAV edge nodes and the cloud server, a multi-UAV enabled edge-cloud system occurs. In the system, UAVs are able to offload the tasks to other UAVs or the cloud server. Several researches have focused on the multi-UAV enabled edge-cloud system. A decentralized computation offloading mechanism for such system was proposed to achieve the energy cost minimization in [16]. Meanwhile, Xu et al., formulated a stochastic optimization problem with the goal of minimizing the energy consumption in cellular-connected multi-UAV edge computing networks [17]. Energy cost minimization is absolutely a key factor to maintain the reliability, while the computation delay minimization should also be considered in the strategy as a basis for judging whether the critical mission is successfully executed [25]. Compared with [16], [17], we focus on the delay optimization while meeting the energy constraints to satisfy the real-time requirements. Although some literatures considered delay as a metric while using multi-UAV to assist the edge-cloud computing [26], [27], UAVs in those literatures were not treated as edge nodes but communication relays. The scenarios of [18] is similar to this study, which considers delay-optimal tasks offloading for UAV-enabled edge-cloud computing systems. However, essential factors (e.g., topology, network congestion, resource reservation based on virtualization) are not considered, making the algorithm infeasible to deploy in practical scenarios.

Consequently, it is foreseen that UAV-enabled edge-cloud computing system will be an important trend for edge computing scenarios due to the significantly improved performance upper bound, the flexible deployment and the wide coverage. However, research on multi-UAV enabled edge-cloud cooperative task offloading is still limited. Especially, the cooperative offloading for delay minimization has not been investigated well in the practical multi-UAV enabled edge-cloud computing. Minimizing the service delay of the UAV-enabled edge-cloud system is still a challenge remains to be handled. To make up the gap, this work aims to reduce the system delay to meet the real-time requirements of delay-critical missions in special scenarios from the perspective of cooperative offloading strategy.

3 System Model

As depicted in Fig. 1, we consider an edge-cloud computing system consisted with a BS/ground station (connected with remote cloud) and U rotary-wing UAVs acting as edge computing nodes. All UAVs are executing delay-critical missions and carrying edge computing units. The UAVs are indexed by $\mathcal{U} = \{1, 2, \ldots, U\}$ and the BS is indexed by 0. We consider a system with an expected running time of T slots, and in each slot there will be a number of new data processing tasks generated.

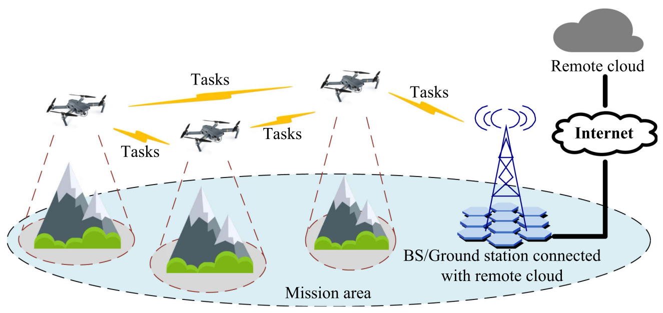


Fig. 1. Multi-UAV enabled edge-cloud computing system.

There are two possible sources of the tasks in practical scenarios. One is ground users/devices. For example, when the desert scientific research mission needs UAVs to enhance computation capability to analyze the complicated data, the ground researchers will generate the tasks and offload them to the nearby UAVs. The other source is UAV itself. Considering the reconnaissance scenario, the UAVs collect large amounts of image and video data for analysis. Thus, UAVs are task generators as well as mission executors. Both sources can be assumed in our model that, the UAV has some tasks to be handled at the beginning of the slot. The length of each time slot is set as τ to depict the real-time requirement of the tasks. In general, a newly generated/received task is considered to be successfully executed only when it is accomplished in the current time slot. However, it is unrealistic for all tasks to be completed in one slot when the number of tasks extremely exceeds the available computing capability. Thereby, the objective for the cooperative offloading is to maximize the probability that tasks can be completed in one slot. Define the delay for UAV i's tasks completion in slot t as *service delay* $D_i[t]$, then the objective is

$$\max_{i \in \mathcal{U}} \Pr\{D_i[t] < \tau\}, \tag{1}$$

τ is set to depict the real-time requirement, i.e., the length of the time slot. The value of τ depends on the feature of the practical mission. In other words, τ can be adjusted according to different missions. However, $D_i[t]$ is a subjective variable, determined by the practical offloading strategy, which means its probability density function is difficult to obtain. Therefore, we adjust the objective from optimizing a probability to an exact value. No matter how large τ is, minimizing the service delay will guarantee the maximal probability for tasks' real-time completion. Thus, the optimization objective can be transformed to an alternative form.

$$\min_{i \in \mathcal{U}} D_i[t]. \tag{2}$$

$D_i[t]$ reflects the individual UAV i's service delay. In order to reflect the overall system service quality, we further define *system delay* as the summation of service delays of all tasks in all time slots. In summary, the objective set as to minimize the system delay

$$\min \sum_{t=0}^{T-1} \sum_{i=1}^{U} D_i[t]. \tag{3}$$

After T slots, the UAV swarm will fly back and terminate the mission. Considering that there is a center controller executing the scheduler algorithm to decide the task-offloading strategy. The center controller may be one of the UAVs or the ground station, who has the global information at the beginning of each slot, i.e., the number of tasks generated and the computation capability of each UAV. Thus, when a new task is received by a UAV, the controller will decide which device is most suitable for offloading, then claim the UAV to offload the task to the selected device. Therefore, a task would experience a two-stage delay before its completion, namely the communication delay and the computation delay. The communication delay is the latency caused by task transmission from a UAV to other devices, while the computation delay is caused by task computation.

$\lambda_i[t]$ and $\pi_{ij}[t]$ are used to describe UAV i's task arrival rate and the number of tasks offloaded to UAV j (or BS) from i in time slot t, respectively, of which $t \in \{1, 2, \ldots, T\}, i \in \mathcal{U}$ and $j \in \mathcal{U} \cup \{0\}$. The tasks are assumed to be divisible, i.e., *partial offloading* is allowed in our work. In other words, $\pi_{ij}[t]$ is not always an integer. For example, when executing video monitoring mission, UAV i has 1,000 frames for an analysis task. Then UAV i has the chance to offload 300 frames to UAV j to alleviate the computation pressure, i.e., $\pi_{ij}[t] = 0.3$. Moreover, we have $\boldsymbol{\lambda}[t] = \{\lambda_1[t], \lambda_2[t], \ldots, \lambda_U[t]\}$, $\boldsymbol{\pi}[t] = \{\boldsymbol{\pi}_1[t], \boldsymbol{\pi}_2[t], \ldots, \boldsymbol{\pi}_U[t]\}$ and $\boldsymbol{\pi}_i[t] = \{\pi_{i0}[t], \pi_{i1}[t], \ldots, \pi_{iU}[t]\}$. Assume that all UAVs have the ability to communicate with the BS, i.e., all tasks received by UAVs can be offloaded to the remote cloud.

UAV's flight trajectory is not considered in our model because the UAVs are assumed to be static to execute the mission. Though the UAVs keep hovering through all time slots, they have the ability to handle most of computation missions, and moving objects can still be handled by static UAVs. Considering that UAVs are monitoring the ground or collect ground sensors' information. Due to the development of high speed camera and the matureness of a variety of sensors born for high speed environment, it is convenient for UAVs to complete the mission. Even in the case of forest fires, UAVs can also take advantage of their camera to obtain infrared images.[3] Considering another scenarios that UAVs serve as base stations in the air to provide wireless link or computing service. The coverage capability and the stability of hovering UAV-BS has been verified by AT&T.[4] The capability can fully meet moving users' service requirements.

The key notations of the model are presented in Table 1 for clarity. Next, we will present mathematical models to describe the delay and energy consumption on communication and computation.

TABLE 1
Summary of Frequency Used Notations

Notation	Definition
$\mathcal{U}$	The set of UAVs carrying EC units $\{1, 2, \ldots U\}$
γ	Coefficient to describe the network congestion
ω	Energy consumption for UAV-to-UAV transmission
$\lambda_i[t]$	Number of UAV i's tasks before offloading
$\beta_i[t]$	Number of tasks offloaded by UAV i at time slot t
$\mu_i[t]$	Number of tasks offloaded to UAV i at time slot t
$\pi_{ij}[t]$	Number of tasks offloaded from i to j at time slot t
P_i	Transmit power of UAV i for communicating with BS
g_u	Power gain of wireless channel at a reference distance
σ^2	The noise power of UAV-to-BS channel
Γ	SNR gap between practical and theoretical signaling
α	The path loss exponent of large-scale fading
$\mathbf{a}_i[t]$	UAV i's horizontal location at time slot t
s	Number of bits to be processed for a task
l	Number of CPU cycles for computing one bit data
D_{BS}	Delay for interaction between the BS and the cloud
ξ	Energy consumption for UAV-to-BS transmission
f_i	CPU frequency of UAV i's EC unit
κ	CPU's parameter depends on chip architecture
$q_i[t]$	The virtual energy deficit of UAV i at time slot t
$\bar{E}_i$	Expected energy budget for UAV i at each time slot

3.1 UAV-to-UAV Transmission Model

Due to the fewer obstacles and the closer distance between UAVs, we assume that the wireless links are Line-of-Sight (LoS) during UAVs interacting with each other. Without loss of generality, we denote R_i as the UAV-to-UAV achievable rate. In addition, the network topology should be considered when modeling the UAV-to-UAV channel. There are two kinds of network topology for UAV-to-UAV transmission, i.e., star network topology and mesh network topology. In the star network topology, a node acting as the hotspot forwards the information to other nodes. In the mesh network topology, nodes relay traffic for other nodes on the path. Mesh [28] has been widely used in fixed-wing aircraft networking and several studies have investigated the rotary-wing UAV mesh networks [29]. However, the mainstream communication protocol for rotary-wing UAV communication, i.e., MAVLink [30], is still based on the control station system to execute the UAV-to-UAV communication, which corresponds to the star network topology. Thereby, in this subsection, star-shape network is used to describe the UAV-to-UAV network. Existing work [31], [32] commonly consider the delay of task transmission among edge nodes as a linear model, i.e., in the Local Area Network (LAN) consists of UAVs, transmission delay is proportional to the number of offloaded task, as Equation (4) depicts.

$$D_{i,Linear}^{LAN} = \beta_i[t]\frac{s}{R_i}. \tag{4}$$

D_i^{LAN} is the transmission delay caused by UAV i offloading its tasks to other UAVs. $\beta_i[t]$ is the number of tasks that be offloaded to other UAVs by UAV i, denoted by $\beta_i[t] =$

3. https://enterprise-insights.dji.com/blog/drones-in-forest-fire-response
4. https://technewsspace.com/att-is-putting-5g-base-stations-on-drones/

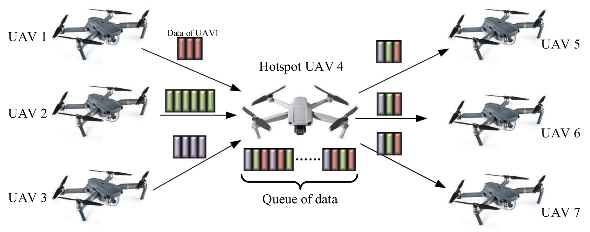


Fig. 2. Queue of data in the star-shape network.

$\sum_{j=1,j\neq i,j\neq 0}^{U}\pi_{ij}[t]$. s is the expected number of bits to be processed for a task. Define $\beta[t]$ as the total number of offloaded tasks by all UAVs during time slot t, which can be described as $\beta[t]=\sum_{i=1}^{U}\beta_i[t]$.

When multiple UAVs offload their tasks concurrently in the star-shape network, congestion occurs because the bandwidth and the buffer are limited, which affects transmission latency critically. Multiple UAVs construct a LAN through the center node (i.e., the hotspot) who realizes the resource sharing. As Fig. 2 depicts, the tasks' data from UAVs form a queueing model in the hotspot, and the hotspot forwards the data to other UAVs basing on the offloading decision. The limited bandwidth and buffer cause the service rate lower than arrival rate, which makes the length of the queue increase in a non-linear pattern [33]. Therefore, congestion which depends on the total traffic in the LAN should be taken into account in the transmission model. Here we only consider the queue at hotspot. There are two main reasons. (i) The star-shape topology determines the hotspot will be the bottleneck link. A queue is formed when the service rate is less than arrival rate. Even a UAV receives tasks from several other UAVs, the traffic on it will not more than the traffic on the hotspot. Thus, the hotspot holds the highest arrival rate. Meanwhile, the service rate of the hotspot is the lowest because it is responsible for forwarding while others only need restore the tasks in local buffer. Therefore, the bottleneck should be the hotspot, which leads to the network congestion. (ii) Compared to sending, receiving tasks is not time-consuming. Existing methods make data receiving efficiently. For example, I/O multiplexing in Linux (e.g., epoll) can monitor the incoming data of multiple ports in batches, and the multi-coroutines are able to handle the jobs for receiving data concurrently. Industrial enterprises have widely adopted the methods (e.g., I/O multiplexing and multi-coroutines) to reduce the delay of receiving significantly. In conclusion, the effect of UAV i's queue delay will not show up in our system.

M/M/1 model [34] was previously adopted by [35], [36] to describe the network congestion in the LAN make up of several wireless devices. The authors assume that the service rate of the LAN is a variable following exponential distribution and the data size of computation tasks offloaded to each device follows the exponential distribution. Nevertheless, most of the modern remote procedure call (RPC) protocols (e.g., Google's gRPC [37]) using in the distributed system, or the commonly used messaging protocol (e.g., MQTT protocol [38]) in M2M (machine to machine) scenario, are both based on TCP protocol. TCP protocol has its own congestion control and avoidance mechanisms. Thus, congestion happens inevitably but the impact will not be as severe as the M/M/1 model due to the mechanisms. TCP alleviates the network congestion at the cost of transmission performance. Thereby, comparing with the linear model, a coefficient γ should be added to reflect the loss of performance.

$$D_i^{LAN}=\gamma\cdot\beta_i[t]\frac{s}{R_i}. \tag{5}$$

Nevertheless, γ is not a constant. As network traffic increases, so does the overhead for controlling congestion. Thereby, it can be considered that γ is a function whose argument is the overall traffic $\beta[t]$

$$\gamma=\gamma_1\beta[t]s+\gamma_2. \tag{6}$$

Therefore, the UAV-to-UAV transmission delay of UAV i in the system can be denoted as

$$D_i^{LAN}=(\gamma_1\beta[t]s+\gamma_2)\frac{\beta_i[t]s}{R_i}. \tag{7}$$

Once a task is offloaded, UAV who receives the task will immediately start computing because of the asynchronous multithreading method [39]. To verify the UAV-to-UAV transmission delay model, measurement has been done on a practical experiment platform. Related results are recorded in Section 5.1.2. The results indicate that the model in Equation (7) is more relevant to reality compared with the linear model and the M/M/1 model.

Define the expected average energy consumption for transmitting a task between UAVs as ω, then in time slot t, UAV i's energy consumption for offloading tasks to other UAVs can be modeled as

$$E_i^{LAN}[t]=\omega\cdot\beta_i[t], \tag{8}$$

ω is an expected value of the average energy consumption for transmitting a task via the LAN. Since all UAVs complete a mission together, the tasks of each UAV are assumed to be isomorphic. Thus, differences in tasks of various UAVs are considered small. Without loss of generality, even if the tasks' sizes are not as the same, the operation of taking expected value will ignore those differences. The idea of taking the expected value is also used in the following subsections.

3.2 UAV-to-Cloud Transmission Model

In recent years, there have been many researches on UAV communications, and the modeling of the UAV-to-Ground channel is relatively mature. So in this subsection, classical UAV-to-Ground wireless channel is adopted to model the transmission delay between UAV edge node and the cloud server when offloading the tasks. Assume that all UAVs hovers at a constant altitude H, BS locates at $\mathbf{a}_{BS}=(x_{BS},y_{BS})$ and UAV i's horizontal location at time slot t is $\mathbf{a}_i[t]=(x_i[t],y_i[t])$. The tasks offloaded by UAVs to the cloud will arrive at the nearby BS then will be forwarded to the remote cloud. The environment between UAV and BS is complicated, both the Line of Sight (LoS) channel and the Non-Line of Sight (NLoS) channel should be considered.

Considering LoS channel, free space transmission loss was used to depict the air-to-ground channel [40]. The channel power gain between the UAV and the BS, denoted by $h_i[t]$, can be given as

$$h_i[t] = g_u d_{i,0}^{-2} = \frac{g_u}{H^2 + \|\mathbf{a}_i[t] - \mathbf{a}_{BS}\|^2}, \tag{9}$$

g_u is the channel power gain at a reference distance $d_0 = 1$m. Thus, the achievable rate of LoS channel can be denoted as

$$R^s_{i,LoS}[t] = B\log_2\left(1 + \frac{h_i[t]P_i}{\sigma^2}\right), \tag{10}$$

B is the bandwidth allocated to UAVs to communicate with BS, P_i is the transmit power of UAV i, and σ^2 denotes the noise power.

Considering NLoS channel, there are tall buildings as obstacles between UAV and BS, and distance is far, so large-scale fading and small-scale fading should be taken into account. In this case, we adopt classical air-to-ground channel model in [41], which uses block fading channels to model the link, so data rate of the link between UAV i and BS can be modeled as

$$R^s_{i,NLoS}[t] = B\log_2\left(1 + \frac{F^{-1}(\epsilon)P_i g_u}{\sigma^2\Gamma\left(H^2 + \|\mathbf{a}_i[t] - \mathbf{a}_{BS}\|^2\right)^{\alpha/2}}\right), \tag{11}$$

$F(\cdot)$ denotes the identical cumulative distribution function (CDF) of small-scale fading block. In this paper, we consider the Rician fading channels with Rician factor K_c, and the CDF function can be expressed by $F(z) = 1 - Q_1(\sqrt{2K_c}, \sqrt{2(K_c+1)z})$ where $Q_1(a,b)$ is the Marcum-Q function, $K_c = 10$ [41]. ϵ means the maximum tolerable outage probability, $\Gamma > 1$ is the SNR gap between the practical modulation schemes and the theoretical Gaussian signaling, $\alpha \geq 2$ is the path loss exponent of large-scale fading.

Both LoS channel and NLos channel should be involved to describe the complicated UAV-to-Ground channel. For this reason, the method proposed by [42] is adopted which regards the UAV-to-Ground channel as the weighted sum of LoS and NLoS channel. The weight is adapted to the terrain environment such as urban, suburban, etc.

$$R^s_i[t] = \Pr(LoS,\theta)\cdot R^s_{i,LoS}[t] + \Pr(NLoS,\theta)\cdot R^s_{i,NLoS}[t]. \tag{12}$$

$\Pr(LoS,\theta)$ is the probability of LoS channel. $\Pr(NLoS,\theta) = 1 - \Pr(LoS,\theta)$ and the probability is given as

$$\Pr(LoS,\theta) = \frac{1}{1 + ae^{-b(\theta - a)}}, \tag{13}$$

a and b are parameters contingent on different environments (e.g., suburban, urban, dense urban). The model predicts the expectation of channel path loss from the perspective of statistics.

The model mentioned above considers several scenarios. However, an essential scenario for UAV's mission execution has not been considerd, i.e., forest. Near Line of Sight (nLoS) has been proposed to depict the air-to-ground channel under the forest scenario. Therefore, we adopted a classical nLoS air-to-ground model to improve our model [43]. The path loss in nLoS channel is denoted as

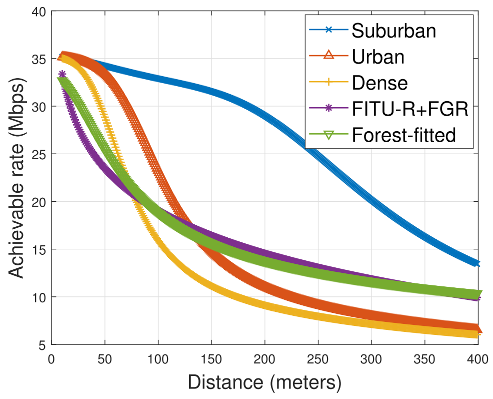


Fig. 3. Achievable rate of UAV-to-cloud link.

$$\begin{aligned} L_{\text{FITU-R}}(\text{dB}) &= 0.39 f^{0.39} d^{0.25} \\ L_{\text{FGR}}(\text{dB}) &= 10n\log_{10}(d) - 20\log_{10}(h_T) - 20\log_{10}(h_R) \\ L_{\text{forest}}(\text{dB}) &= L_{\text{FITU-R}} + L_{\text{FGR}}. \end{aligned} \tag{14}$$

The model is named as "FITU-R+FGR". f is the frequency of the radio wave (in MHz). d is the distance between the isotropic transmit and receive antennas in meters. n is a parameter related to f. h_T and h_R are the height of transmitter and receiver, respectively. However, we found that Eq. (12) actually includes the "FITU-R+FGR" model. nLoS can be approximately considered as an intermediate state between LoS and NLoS. Through fitting the model, $a = 4.0510374, b = 0.0478963$ are set. When setting the parameters, the rate that Eq. (12) achieves is almost identical to the rate "FITU-R+FGR" achieves, as Fig. 3 shows. In Fig. 3, the curve 'Forest-fitted' is the results when setting a and b in Eq. (12), with $P_i = 0.1$ Watts assumed. Therefore, the achievable rate of UAV-to-Cloud channel in our system can still be set as Eq. (12). The parameters $a = 4.0510374$ and $b = 0.0478963$ are needed when the scenario is in forest.

In addition, Remote cloud is supposed to be far from the BS. Thus, when the BS forwards tasks to the cloud through a wired link, latency is inevitable. The latency caused by one task offloaded from UAV to the cloud can be denoted as

$$D^d_i[t] = \frac{s}{R^s_i[t]} + D_{BS}, \tag{15}$$

$s/R^s_i[t]$ is the delay while offloading a task from UAV i to the BS, and D_{BS} is the delay for exchanging data (bits to be processed and the computation results) between the BS and the cloud. It is necessary to set the D_{BS}. Though BS forwards the tasks to the remote cloud exchanges data through wired link, there is still high latency due to the distance. Using *Ping* command to measure the Round-Trip Time (RTT) between local host (Wuhan City) and a cloud server (Hong Kong), the average latency is 200.625 ms (with the distance of 1105 km). Due to the fact that data amount of computation results is tiny, delay caused by feedback from the BS to the UAV is not considered.

Furthermore, define the expected average energy consumption for transmitting a task from a UAV to the BS as ξ, then UAV i's energy consumption for offloading tasks to the cloud is

$$E^d_i[t] = \xi \cdot \pi_{i0}[t]. \tag{16}$$

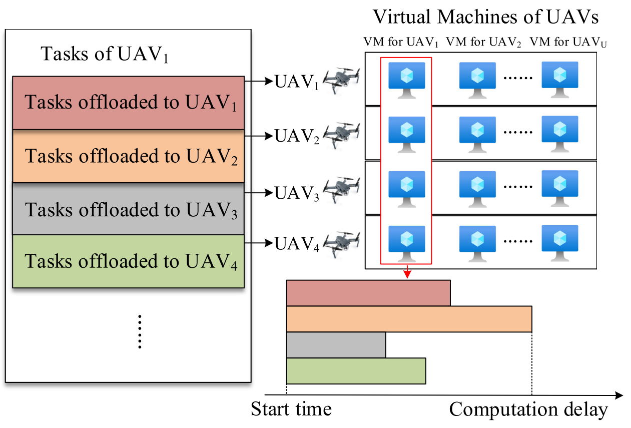


Fig. 4. Tasks of UAV1 computed by the UAV cluster.

3.3 Computation Model

The computation time of the cloud is neglected since it has much stronger computation capability than the UAV-mounted EC unit. The energy consumption of the cloud is not involved in the multi-UAV system. Thereby, in this section, only the delay and the energy consumption caused by UAV's computation are taken into account. Define the CPU frequency of each UAV-mounted EC unit, as $\mathbf{f} = \{f_1, f_2, \ldots, f_U\}$. It is assumed that each EC unit has multiple cores.

To overcome the incompatibility when executing the programs in UAVs with heterogeneous specs, the UAV cluster (formed by U UAVs) is assumed to leverage the virtualization technology or containerization technology [44]. The virtual machines within a UAV are prepared to complete the tasks from other UAVs. For example, UAV i's virtual machine, which prepares for UAV j, will remain idle and hardly occupy the CPU until j's tasks offloaded to i. Assuming that each virtual machine or container has the ability to occupy single thread, a UAV can compute the tasks from different UAVs concurrently, and tasks for single virtual machine are still executed in sequence. As illustrated in Fig. 4, the task of UAV 1 is divided into four subtasks, each subtask is offloaded to the corresponding UAV j which is represented by the corresponding color. Owing to Time Division Multiplexing mechanism and the fact that computation delay is much longer than communication delay, the starting point of different virtual machines (or containers) in UAV i for tasks computation can be considered almost simultaneously (the validity of the assumption is verified in Section 5.1.3.), then the completion time of whole task depends on the slowest subtask of the computation.

Considering the multi-UAV edge-cloud system as a distributed system, the tasks offloaded by UAV i are executed in parallel by other UAVs but not in sequence. Thus, the computation delay of UAV i's tasks in the slot depends on the longest delay among the devices, as Fig. 4 depicts. For this reason, computation delay for all tasks offloaded by UAV i in time slot t can be modeled as

$$D_i^{cal}[t] = \max_{j\in\mathcal{U}} \left\{ \pi_{ij}[t] \frac{s \cdot l}{f_j} \right\}, \tag{17}$$

l is the expected number of CPU cycles to compute 1 b data. The verification of the computation delay model can be found in Section 5.1.3, and the verification results indicate the model is corresponding with the practical scenario.

Denote $\mu_i = \sum_{k=1}^{U} \pi_{ki}[t]$ as the number of tasks that offloaded to UAV i by other UAVs at time slot t. Energy consumption caused by computation is proportional to the square of CPU frequency [2], so UAV i's energy consumption for computation at time slot t is denoted by

$$E_i^{cal}[t] = \kappa f_i^2 \cdot \mu_i[t], \tag{18}$$

κ is the parameter depends on effective switched capacitance of CPU's chip architecture and computation task itself.

4 Problem Formulation and Solution

4.1 Problem Formulation

In our assumption, a UAV is unable to communicate with several UAVs and the BS simultaneously due to single-antenna. Therefore, we assume that each UAV occupies the channel in a CSMA (Carrier Sense Multiple Access) manner during UAV-to-UAV communication. Meanwhile, multiple UAV-to-BS links are in time-sharing manner, i.e., TDMA (Time Division Multiple Access). Thus, different UAVs within the LAN can offload the tasks to each other in concurrently, while the tasks from a UAV to the cloud are sent in order. More specifically, multi-hop communication is still in the form of summation. From the above, UAV i's service delay (i.e., delay to accomplish all tasks generated/received by i in time slot t) can be donated by

$$D_i(\boldsymbol{\pi}[t]) = D_i^{LAN}[t] + D_i^{cal}[t] + \pi_{i0}[t] D_i^d[t], \tag{19}$$

and its energy consumption is

$$E_i(\boldsymbol{\pi}[t]) = E_i^{LAN}[t] + E_i^{cal}[t] + E_i^d[t], \tag{20}$$

$E_i(\boldsymbol{\pi}[t])$ contains two kinds of energy consumption. One is caused by computation for the tasks offloaded by all UAVs (other UAVs and itself), while the other energy consumption is caused by transmitting part of tasks to other devices (other UAVs or the BS).

The purpose is to minimize the system delay within the battery budget. Problem can be formulate as follow:

$$\begin{aligned}
&\mathbf{P1:} \min_{\boldsymbol{\pi}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{U} D_i(\boldsymbol{\pi}[t]) \\
&s.t.\ C1: \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E_i(\boldsymbol{\pi}[t]) \le \bar{E}_i, \forall i \in \mathcal{U} \\
&\quad C2: \pi_{ij}[t] \ge 0, \forall i \in \mathcal{U}, \forall j \in \{0\} \cup \mathcal{U}, \forall t \\
&\quad C3: \sum_{j=0}^{U} \pi_{ij}[t] = \lambda_i[t], \forall i \in \mathcal{U}, \forall t \\
&\quad C4: \mu_i[t] \le \frac{f_i T}{sl}, \forall i \in \mathcal{U}, \forall t.
\end{aligned} \tag{21}$$

Constraint $C1$ means UAV's energy consumption cannot violate the long term energy constraint, $\bar{E}_i$ is the expected energy budget for UAV i at each time slot. $C2$ avoids invalid values in cooperative offloading decision. $C3$ guarantees that tasks offloaded from UAV i is equal to the tasks generated by the UAV, and $C4$ guarantees that the tasks offloaded to UAV i will not exceed its computation capability.

The energy consumption for UAV hovering is not considered in our system. The system does not involve energy sharing between flight control modules (e.g., motors) and edge computing modules (e.g., communication and computation modules) because we utilize two separate power supply. Through the separate power supply, energy consumed by edge computing will not affect the power supply for flight control, and several hazards (e.g., overcurrent damage) can also be avoided. Meanwhile, formulating hovering energy consumption will not affect the problem. Assuming the power for maintaining hovering is $\mathcal{K}$ Watts and the length of a time slot is $\mathcal{X}$ seconds, energy consumption caused by hovering can be seemed as a fixed value $\mathcal{K} \cdot \mathcal{X}$. The fixed value will not affect the correction of constraint $C1$. Therefore, the energy budget in this study is prepared for edge computing but not flight control. And in the problem $\mathcal{P}1$, the energy consumption for hovering is not considered.

4.2 Lyapunov Optimization-Based Approach

In each slot, the number of tasks arrived at UAVs is supposed to be random. Therefore, it is infeasible to know the exact number of tasks at the beginning of system operation. In this section, we transform the global optimization problem into an online optimization problem by Lyapunov optimization method. Then, convex approximation is utilized to make the non-convex online optimization problem tractable. The approximated problem can be handled by existing convex optimization tools.

1) Transformation to Online Problem: The Lyapunov optimization method can make a tradeoff between system delay and the energy consumption under the premise of unknown number of tasks generated in future time slots. Original problem can be transformed to Lyapunov optimization form as follow using the *Lyapunov drift-plus-penalty technique* [45]:

$$\mathbf{P2}: \min_{\boldsymbol{\pi}} \sum_{i=1}^{U} \{V \cdot D_i(\boldsymbol{\pi}[t]) + q_i[t] \cdot E_i(\boldsymbol{\pi}[t])\} \quad s.t.\ C2, C3, C4, \tag{22}$$

$\boldsymbol{q}[t] = \{q_i[t]\}_{i \in \mathcal{U}}$ are the virtual energy deficit queues, let $q_i[0] = 0, \forall i \in \mathcal{U}$. $q_i[t]$ denotes the queue length in time slot t, which evolves as follow:

$$q_i[t] = \max\{q_i[t-1] + E_i(\boldsymbol{\pi}[t-1]) - \bar{E}_i, 0\}. \tag{23}$$

Theorem 1. Consider D_L^* as the optimal value of problem **P2**, and D^* as the optimal value of problem **P1** with the global information, i.e., the theoretical lower bound. Then the gap between D_L^* and D^* is $O(\frac{1}{V})$. And the time-average length of virtual energy deficit queues are $O(V)$.

Proof. See Appendix A, available online. □

The theorem above shows that, as the parameter V increases, the optimal solution obtained by the Lyapunov optimization will get closer to the theoretical lower bound, yet the length of the energy deficit queue will increase accordingly, resulting in ascending energy consumption.

2) Convex Approximation: Discontinuity of the objective function still exists in problem **P2**, making it hard to solve using traditional algorithm like Newton Method. To make the problem tractable, the discontinuity part, i.e., $\max(\cdot)$ function, should be approximated to convex form.

Theorem 2. Given a large enough parameter η, computation delay $\max_{j\in\mathcal{U}}\{\frac{\pi_{ij}}{f_j}\}$ can be approximated to $\frac{1}{\eta}\lg(\sum_{j=1}^{U} e^{\eta\pi_{ij}/f_j})$

Proof. [46] shows that $\max(\cdot)$ function meets the inequality $\max\{x_1,\ldots,x_n\} \le \lg(\sum_{i=1}^{n} e^{x_i}) \le \max\{x_1,\ldots,x_n\} + \lg n$, thus, through introducing parameter η, $\max_{j\in\mathcal{U}}\{\frac{\pi_{ij}}{f_j}\}$ satisfies

$$\begin{aligned}\max_{j\in\mathcal{U}}\left\{\frac{\pi_{ij}}{f_j}\right\} &= \frac{1}{\eta}\max_{j\in\mathcal{U}}\left\{\eta\frac{\pi_{ij}}{f_j}\right\} \\ &\le \frac{1}{\eta}\lg\left(\sum_{j=1}^{U} e^{\eta\pi_{ij}/f_j}\right) \\ &\le \max_{j\in\mathcal{U}}\left\{\frac{\pi_{ij}}{f_j}\right\} + \frac{1}{\eta}\lg U.\end{aligned} \tag{24}$$

□

Consequently, $\lim_{\eta\to\infty}\frac{1}{\eta}\lg(\sum_{j=1}^{U} e^{\eta\pi_{ij}/f_j}) = \max_{j\in\mathcal{U}}\{\frac{\pi_{ij}}{f_j}\}$. With η being larger, degree of approximation will be better, so we can define a parameter η and convert the original function to the form of *log-sum-exp* function, while the latter is convex.

Algorithm 1. LOC: Lyapunov-Based Online Cooperative

```
Input: $V$, $\eta$
Output: Offloading decision $\boldsymbol{\pi}$
1: Initialize energy deficit queue $q_i[0] = 0, \forall i \in \mathcal{U}$
2: for $t = 0$ to $T-1$ do
3:   Count the arrival rate of each UAV $\lambda_i[t]$
4:   Solve P3 by SCS solver
5:   $q_i[t+1] = \max\{q_i[t] + E_i(\boldsymbol{\pi}[t]) - \bar{E}_i, 0\}, \forall i \in \mathcal{U}$
6: return $\boldsymbol{\pi}$
```

Now, we can transform problem **P2** to convex optimization problem **P3**, which can be solved by optimization tools. Based on the above derivation, the Lyapunov optimization-based Online Cooperative (**LOC**) task offloading algorithm is proposed. Before system running, the energy deficit queue of each UAV should be initialized. At each time slot, problem **P3** is solved by SCS solver[5] which calculates the optimal solution through ADMM approach, and then the energy deficit queue would be updated. The process should be repeated from one slot to another until the last. The algorithm is executed by the central controller of UAVs, who has a global view of each UAV' s state at the beginning of each time slot.

$$\begin{aligned}\mathbf{P3}: \min_{\boldsymbol{\pi}} \sum_{i=1}^{U}\Bigg[V \cdot \Bigg(\frac{s\cdot l}{\eta}\lg\Bigg(\sum_{j=1}^{U} e^{\eta\pi_{ij}[t]/f_j}\Bigg) + \pi_{i0}[t]D_i^d[t] \\ + D_i^{LAN}[t]\Bigg) + q_i[t]\cdot E_i(\boldsymbol{\pi}[t])\Bigg] \\ s.t.\ C2, C3, C4.\end{aligned} \tag{25}$$

5. https://github.com/cvxgrp/scs

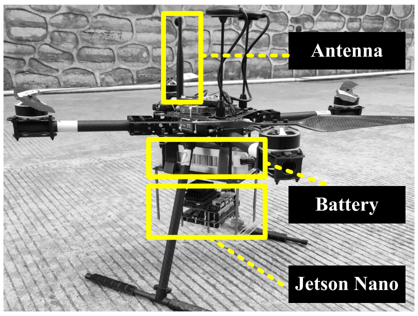


Fig. 5. UAV-enabled edge computing platform.

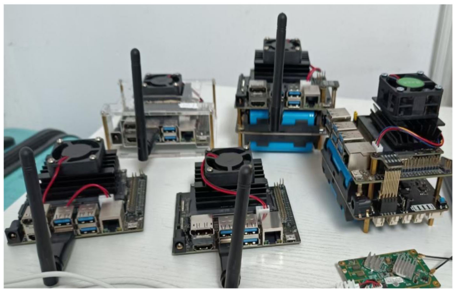


Fig. 6. Jetson Nano simulating a UAV swarm.

The choice of solver is not unique. Actually, all solvers that has the ability to solve the convex problem are suitble to LOC. We choose the SCS solver because it is free, efficient and flexible.

The key parameters in Eq. (7), i.e., γ_1 and γ_2, will be given later in Section 5.1.2. We have measured the congestion-related parameters in our testbed. However, in practical scenarios, the parameters may need to be adjusted. Therefore, a method is also given later to guide how to obtain γ_1, γ_2 in practical scenarios.

5 Performance Evaluation

This section focuses on the performance evaluation. Section 5.1 introduces the construction of our UAV-EC experimental platform, and utilizes the practical platform to verify the correctness of the proposed model. After the model verification, Section 5.2 will implement simulation experiments based on real-world datasets, to explore the performance of LOC algorithm from different perspectives.

5.1 Model Verification Through a UAV-EC Platform

5.1.1 UAV-Enabled Edge Computing Platform

A UAV-enabled edge computing platform is presented as Fig. 5. Using Pixhawk4[6] as the flight control unit, a rotary-wing UAV is assembled. The edge computing unit which undertake the obligation of computation is Jetson Nano.[7] Jetson Nano is a computer with a GPU produced by Nvidia and the GPU makes it be able to perform deep learning inference tasks as a lightweight, low-power edge computing node. T208[8] is used to guarantee the power supply of Jetson Nano. MorningCore's dedicated UAV star network transmission module[9] is chosen to exchange data among the UAVs. After selecting the appropriate motor (AXI 4120 brushless motor), the UAV edge node is able to fly smoothly and hover stably as a powerful computation unit. The components selected have been widely used in existing UAV systems. For example, Pixhawk is the hardware standard for open-source autopilots and it is the world's most popular open source flight controllers available, and Jetson has been used to deploy a wide range of popular DNN models and ML frameworks to the edge with high performance inferencing.[10] MorningCore's module is also an industrial product which has a certain market share in China.

Due to the field area restriction and the safety regulations, we cannot assemble too many UAVs and fly them in a wide space for testing. Therefore, we measure the achievable data rate between two UAV edge nodes, then use several Jetson Nanos at the ground to simulate the UAVs on the air (Fig. 6). Two UAV-enabled edge nodes were constructed and the achievable rate between them is measured using iperf[11] tool when they are flying. Then, using tc[12] provided by Linux, the maximum data rate among Jetson Nanos at the ground is restricted to imitate the situation in the air. We performed the measurements between two UAVs in an open environment, as Fig. 7 shows (marked with Google Maps). Then, we repeated the measurements for 10 times, and the average results is recorded in Fig. 8. It can be observed that, the achievable rate is always higher than 15 Mbps when the distance is smaller than 400 meters. Thus, to simulate the worst wireless communication condition literally in the fly, the average data rate among Jetson Nanos at the ground is restricted to 15 Mbps. Then the nodes are able to simulate a UAV-enabled edge computing cluster.

Apart from that, we have utilized the UAV-EC platform to measure an air-to-ground channel. The measured results are leveraged to imitate the practical UAV-to-BS c
hannel. Keeping a UAV-EC platform at the height of 20 meters, we enable the communication between it and the equipment on the ground. Setting the transmission power as 0.2 W, the achievable rate between UAV and the ground equipment is measured, and the results are recorded in Fig. 9. Then, we explore the consistency between measured rate and our model, i.e., Eq. (12). Fitting the parameters, we found that our model is conform to the reality when $a = 2.05983, b = 0.44375$. We measured the air-to-ground channel in campus environment, whose rate should be between urban and suburban, and the results are in line with the theoretical expectation.

It is worth mentioning that, MorningCore's modules on the UAVs request the topology of the network to be the shape of star, i.e., there is a hotspot among them to organize a Local Area Network (LAN). Meanwhile, in the ground system shown in Fig. 6, the devices are connected within a LAN and there is also a hotspot to make the system's topology as star shape. Thereby, the topology of the ground Jetson

6. https://docs.px4.io/master/en/flight_controller/pixhawk4.html
7. https://elinux.org/Jetson_Nano
8. https://wiki.geekworm.com/T208
9. https://www.morningcore.com/site/products_info/91
10. https://developer.nvidia.com/embedded/jetson-benchmarks
11. https://iperf.fr/
12. https://man7.org/linux/man-pages/man8/tc.8.html

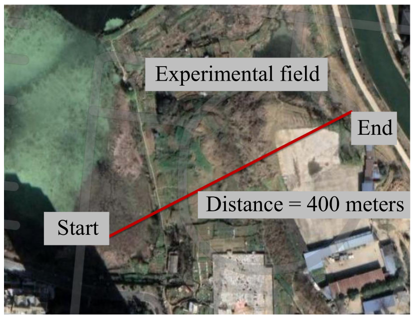


Fig. 7. Environment for measurement.

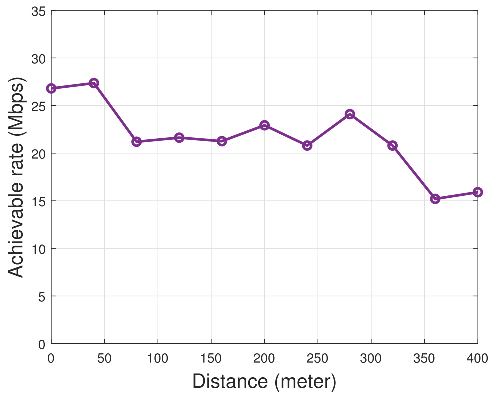


Fig. 8. Achievable rate between two UAVs.

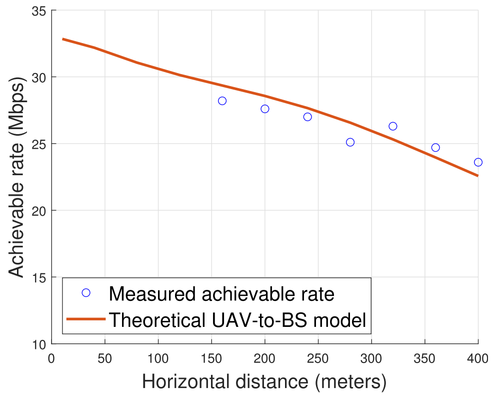


Fig. 9. Air-to-ground Achievable rate.

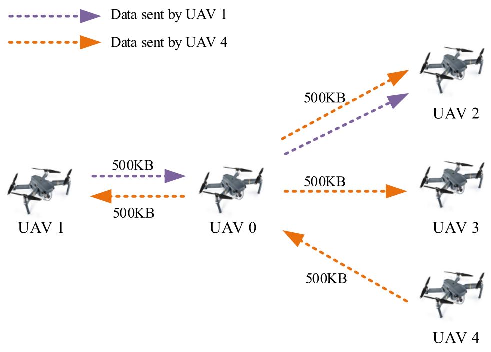


Fig. 10. Experiment designed to verify the proposed UAV-to-UAV model.

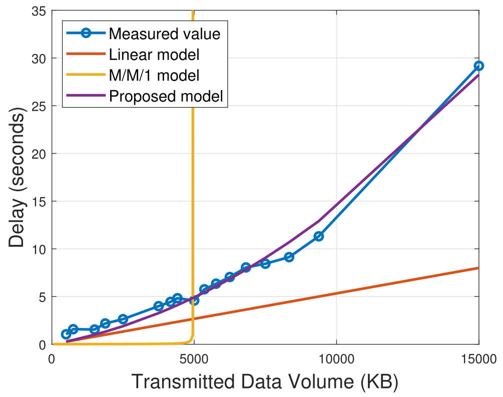


Fig. 11. Different models comparing with measured result.

Nanos' system is keep unchanged compared with the UAVs in the air. The hotspot becomes the bottleneck of UAV-to-UAV transmission when large amount of data is offloaded. Network congestion occurs due to the bottleneck [47].

5.1.2 Verification for UAV-to-UAV Transmission Model

The verification experiment is designed as Fig. 10. UAV 0 is the hotspot, UAV 1 sends data to UAV 2 through the hotspot with the data amount changing from 500 KBytes to 15 MBytes. Meanwhile, there is an another node (i.e., UAV 4) sending data to all nodes except itself. The amount of data sent to other nodes by UAV 4 are the same as the amount sent by UAV 1. Thus, UAV 1 and UAV 4 compete the network resource in the star-shape network with UAV 1 as the bottleneck. According to Section 5.1.1, the average data rate of each UAV is restricted to 15 Mbps. The transmission delay of UAV 1 was measured to observe the most suitable congestion model of the UAV cluster.

Measurement results are recorded in Fig. 11. Measured values are the transmission delay for offloading UAV 1's data under the congestion. Traffic amount means the amount of data sent by UAV 1, according to the $\beta_i[t] \cdot s$ item in Equation (7). The amount has the range from 500 KBytes to 15 MBytes. Fitting the Equation (7) through least squares method, it can be observed that the proposed model is much more related to the measured results than linear model and M/M/1 model. In other words, the proposed UAV-to-UAV transmission model is more relevant to the reality. γ_1 and γ_2 are calculated as 4.2198976×10^{-9} and 1.022729, respectively.

Two key prerequisites are met in real world multi-UAV system: (i) there is a hotspot as bottleneck within the cluster, and (ii) there is a central controller to control the entire system. The central controller is able to trigger the bandwidth probe, and the hotspot is definitely the cause of congestion. Thus, if γ_1 and γ_2 are tended to be modified according to the practical scenarios, the framework for the verification experiment is suitable. An algorithm is provided for reference in appendix B, available in the online supplemental material. The algorithm refers to BBR algorithm [47], and could be used to obtain γ_1 and γ_2 in practical scenarios.

5.1.3 Verification for Computation Model

Docker[13] has been deployed in the test platform as the containerization technology to observe whether the computation

13. https://www.docker.com/

delay model fits the practical application. In this subsection, we select object detection as the computation tasks among the UAVs. Object detection is a typical task for UAV in reconnaissance, search, rescue or other missions. YOLOX [48] is a novel object detection algorithm which has the ability to achieve accurate detection results. In our work, YOLOX is packaged as docker image and each node (i.e., Jetson Nano) has the image. During the system initialization, the container will be constructed through the image to execute the object detection. Five frames of a video with the size of 1280x720 are sent to the nodes concurrently, and each picture accords to one node. Once a node finishes its job, the coordinate of the object detected will be send back in form of text for integration. Thus, the amount of task data transmitted by the initiator (i.e., the node generate the tasks) is far more than the computation result received. In order to make the verification results more convincing, GPU in each node has been disabled and the available CPUs for the task of 5 nodes are restricted to $\{0.2, 0.5, 0.5, 0.8, 1.0\}$ cores by cgroups[14] tool (The maximum value of available CPU is 4.0 because there are four cores in the Jetson Nano's processor). The achievable rate among nodes is set to 15 Mbps to simulate the worst communication condition according to Section 5.1.1.

The timestamp of each node is recorded in Table 2. *Start time* in the table means the node started to execute the task at that time. Once the task data has been received completely, the node begins the computation. The *Start time* column shows the starting point of several containers is nearly the same, which fits our assumption mentioned in Section 3.

Finish time means the time when the task had been accomplished. At that time, the detection result was sent back. The initiator needs all computation results to integrate, so the computation delay depends on the slowest node. In this case, the slowest node is Node 1, who has only 20% CPU available. In Table 2, under the condition of the nearly same starting point, tasks' finishing point is the finish time of Node 1, which is in agreement with the $\max\{\cdot\}$ item in Equation (13).

Time cost is the difference between *Start time* and *Finish time*. From the *Time cost* column, it can be observed that task execution delay is approximately inversely proportional to the CPU frequency, fitting the $s \cdot l/f_j$ item in Equation (13). To make the results for *Time cost* more convincing, 20 more measurements have been done, and the average time cost for all nodes are recorded in Table 3. The average measurement results still fit the Equation (13). In conclusion, the test of the real computation task on practical platform, indicates that our computation delay model (16) is consistent with reality.

5.2 Data-Driven Simulation Experiments

Section 5.1 has verified that the mathematical model is relevant to reality. Thereby, the proposed algorithm based on the model can be easily applied to real-world scenarios. However, several factors result in the difficulty for the algorithm's practical performance test. The factors involve experiment field restriction and safety regulations. Thus, in this section, simulation experiments would be carried out to evaluate the performance of proposed algorithm in multi-UAV enabled edge-cloud systems. The proposed algorithm is compared with several strategies in terms of system delay and energy consumption. To make the simulation as realistic as possible, two real-world datasets have been adopted to imitate the real-world tasks' generation.

TABLE 2
Computation Delay of the Edge Cluster Platform

Node	Available CPU(%)	Start time	Finish time	Time cost
1	20%	07:50:43.32	07:53:24.92	161.60 s
2	50%	07:50:42.12	07.51:40.97	58.85 s
3	50%	07:50:43.10	07:51:43.33	60.23 s
4	80%	07:50:42.03	07:51:21.82	39.79 s
5	100%	07:50:43.33	07:51:17.92	34.59 s

5.2.1 Dataset Description

The two real-world datasets [49] has been frequently adopted by cluster computing system researches for the workloads analysis [50], [51]. The datasets report network traffic data (in bits) time series from two different ISPs, denoted as A and B. The essence of the computation tasks offloaded by UAVs is network traffic. Actually, all the data ISP received can be considered as tasks to be executed because the data is the trigger of the tasks. The size of network traffic can be considered as s times than the number of tasks. Thus, we use the ISP's traffic time series to imitate the tasks generated/received by UAVs at all time slots and scale down the value to a reasonable order of magnitude. The dataset A belongs to a private ISP with centres in 11 European cities. The data was collected from 06:57 hours on 7 June to 11:17 hours on 29 July 2005. Dataset B comes from UKERNA (United Kingdom education and research networking association) and represents aggregated traffic in the United Kingdom academic network backbone. It was collected between 19 November 2004, at 9:30 hours and 27 January 2005, at 11:11 hours. Both the datasets were collected by SNMP scripts and were recorded every 5 minutes.

5.2.2 Experiment Setup

Assume that the whole system is deployed in a $1000\mathrm{m} \times 1000$ m area and the BS locates at $\mathbf{a}_{BS} = (0, 0)$. The number of UAVs was drawn from a random sampling, which followed Poisson distribution with parameter $\lambda = 10$ and the

TABLE 3
Average Time Cost From Multiple Measurements

Node	Available CPU(%)	Time cost
1	20%	158.36 s
2	50%	62.77 s
3	50%	63.12 s
4	80%	38.64 s
5	100%	31.85 s

14. https://man7.org/linux/man-pages/man7/cgroups.7.html

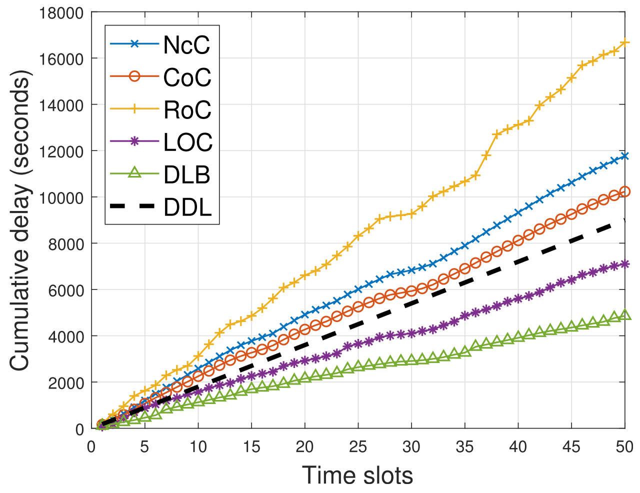


Fig. 12. System cumulative delay under different strategies.

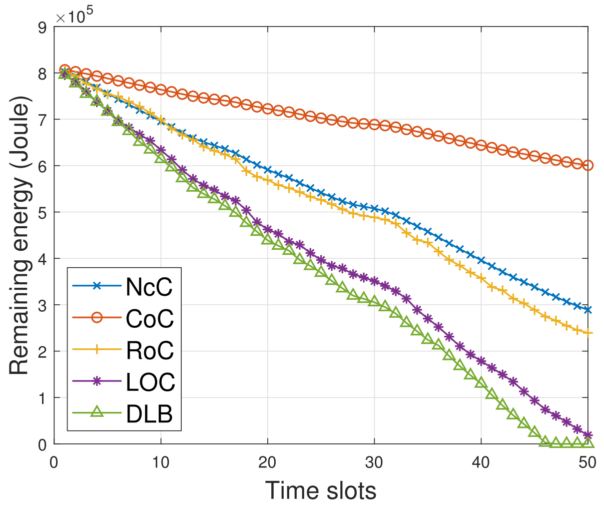


Fig. 13. System remaining energy under different strategies.

sampling result is 9. The horizontal positions of UAVs are randomly drawn from the area following uniform distribution. Once sampled from the uniform distribution, UAVs' positions keep still through all time slots. The height of UAVs is fixed as 50 m. The whole time slice is split to 50 slots and the length of each slot is 3 minutes. Nine subsequences of length 50 are randomly drawn (following uniform distribution) from dataset A and B, to describe the tasks generated by the UAVs. The maximum value of the subsequences is scaled down to 20, i.e., each UAV will generate up to 20 tasks from per slot. To avoid the uncertainty of the experiment result, simulation experiment will be repeated 1,000 times. And the mean values of the 1,000 experiments are reported.

Assume the CPU carried by each UAV has 4 cores, and the CPU frequency f_i is taken from $[0,5]$GHz randomly from uniform distribution. The available energy each UAV has is 9×10^4 J, i.e., the average power is 10 W which is a typical supply power of an onboard edge node [52]. Then $\bar{E}_i$ can be calculated as $\bar{E}_i = \frac{9*10^4}{50} = 1800$ J. $\xi = 30$ J/per task, i.e., transmitting a task to the BS will consume 30 J energy. The same task will cost $\omega = 10$ J when transmitting it to other UAV. Delay caused by long-distance wired communication is $D_{BS} = 200$ ms. Here, we consider the face recognition application in [53], where the computation task has $(3000, 500)$ with the units of Megacycles and KB, i.e., $s = 0.5$ MB. Energy consumption for one bit computation is $\kappa = 10^{-25}$J/bit. Parameters to describe the air-to-ground channel are set as: $g_u =$ -60 dB, $B = 1$ MHz, $P_i = 0.1$ W, $\sigma^2 =$ -110 dBm, $\Gamma = 7$ dB [41].

The performance of LOC algorithm is compared with four benchmarks: *1) Non-cooperative Computing (NcC)*: UAV in the system would execute all tasks locally. *2) Cloud offloading Computing (CoC)*: All tasks would be forwarded to the remote cloud. *3) Random offloading Computing (RoC)*: Each UAV choose random part of tasks to offload to a random device. *4) Delay Lower Bound (DLB)*: Infinite energy consumption is allowed to achieve the system delay lower bound. The difference from LOC is that it ignores the energy queue in each time slot, reflecting the upper bound of the system's computation capability for scheduling. The simulation experiments are implemented in Python 3.9 and performed on a laptop with AMD Ryzen 7 4800 U CPU and 16 GB RAM.

5.2.3 Numerical Results

First, setting $V = 10^7$, $\eta = 10^5$ and terrain environment as urban (i.e., $a = 9.61, b = 0.15$ in the Equation (13)), we explore the performance on system delay and energy consumption of proposed algorithm in Figs. 12 and 13, respectively. *Cumulative delay* is the service delay cumulated from previous time slots of all UAVs. Define cumulative delay as D_{cum} and service delay of each slot as D_{serv}, then $D_{cum}[t] = D_{serv}[0] + \cdots + D_{serv}[t-1] + D_{serv}[t]$. *Remaining energy* is the energy available for future computation or transmission, i.e., Remaining energy = Energy budget - Consumed energy. It is worth mentioning that, the energy budget means the energy can be used for computation and transmission but not for hovering or flying. When the energy consumed equals the budget, offloading and computation will not be allowed but UAVs will still have enough energy to fly back.

"DDL" in Fig. 12 means the cumulative completion deadline of each slot. The length of each slot is 3 minutes, so the ith slot's cumulative deadline is $i\times 3\times 60$ seconds. When a strategy's cumulative delay is above the "DDL," it is indicated that the strategy would time out. In Fig. 12, LOC achieves the minimum cumulative delay compared to other strategies except DLB. But in Fig. 13, remaining energy attracts an attention that DLB causes mission terminated prematurely due to insufficient energy (The delay of DLB after energy exhaustion is still presented in Fig. 12 to reflect the difference more intuitively). Other three strategies leave lots of energy unused. As mentioned in Section 2, UAVs will terminate the mission after T slots. Thus, saving energy is unnecessary for the system because energy for flying has been reserved in advance. A better algorithm should fully utilize the available energy to reduce the delay. From Figs. 12 and 13, it is obvious that only LOC has the ability to realize it. Compared to non-cooperative task execution (i.e., NcC), the proposed algorithm can significantly reduce the system delay by 41.7% according to the Fig. 12.

Accordingly, adjusting the length of a slot τ from 2.5 minutes to 4.0 minutes, the success rate for real-time task completion is reported in Fig. 14. We define the success rate as the ratio of tasks accomplished in the current time slot. As Fig. 14 shows, when τ is large, the success rates of LOC,

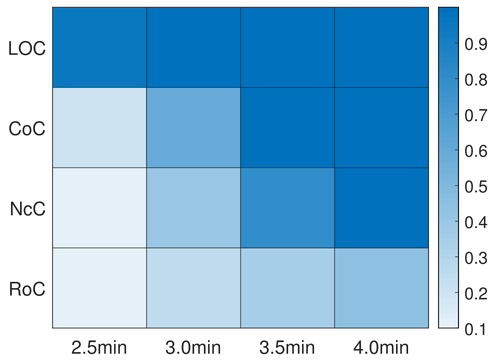


Fig. 14. Success rate under different τ.

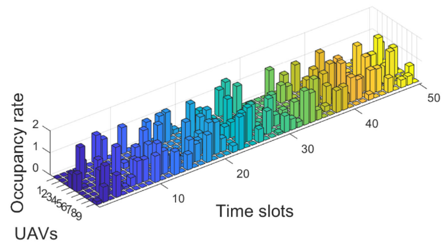


Fig. 15. Load occupancy rate of NcC.

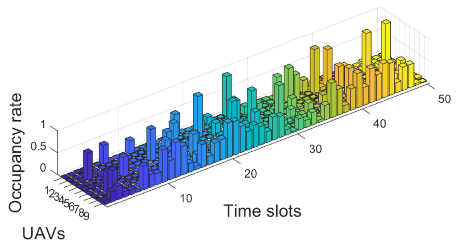


Fig. 16. Load occupancy rate of LOC.

CoC and NcC are high. However, when τ decreases, i.e., the real-time requirement is more strict, only LOC has the ability to maintain a high success rate. Therefore, LOC has the ability to cope with more stringent real-time requirements. And the objective that minimizes system delay is effective for elevating the success rate. When deploying on the practical scenarios, LOC will process the complicated data in real time with higher probability.

Next, the computation load management ability of LOC can be observed from Figs. 15 and 16. We select one of the 1,000 experiments' results randomly, and enumerate the load of all UAVs in all time slots. Load occupancy rate is the ratio calculated by $\frac{\text{Number of tasks to be computed locally}}{f_i/(s\cdot l)}$, which means how much CPU resources have been occupied. In Fig. 15, the rates sometimes exceed 1, which indicates the UAVs endured tasks beyond their capability. Then, the UAVs will be unable to accomplish the tasks in the current slot. In other words, executing all the tasks locally will not satisfy the real-time requirements. But in Fig. 16, the rate is always below 1 which means LOC can make an adjustment to avoid overloading situation happening. Therefore, it is necessary to utilize the LOC algorithm rather than execute the computation locally to assure that the tasks can be accomplished in time.

Then, whether LOC has the ability to adapt different terrain environment is investigated. Three environments (suburban, dense urban and Highrise urban) are tested by adjusting a and b of Equation (13). As Fig. 17 depicts, comparing to the CoC, whose efficiency depends on the air-to-ground channel critically, drastic changes have not happened on LOC.

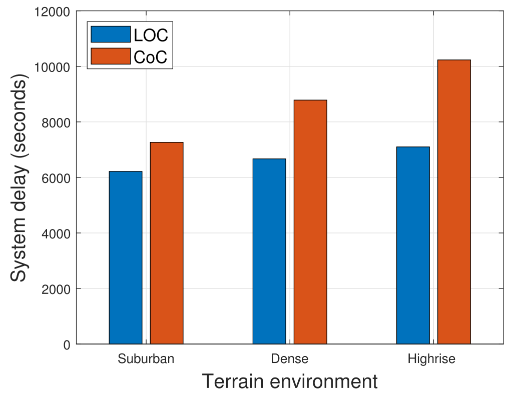


Fig. 17. Adaptability to different environments.

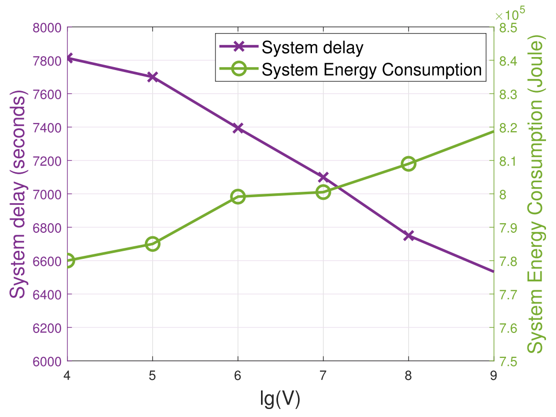


Fig. 18. Impact of parameter V.

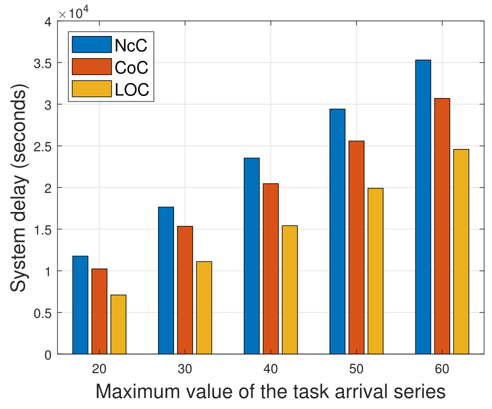


Fig. 19. System delay under increasing number of tasks.

In other words, though the terrain environment changes, LOC can still achieve stable performance in terms of system delay. The results illustrate that LOC has strong adaptability to the environment. LOC will allocate system resources rationally according to the communication condition to maintain stable system delay.

After that, the role of parameter V has also been explored in Fig. 18. Adjusting V from 10^4 to 10^9, the system delay reduced remarkably while energy consumption has an observable increment. The phenomenon is consistent with Theorem 1 (Section 4.2). As V becomes larger, the system

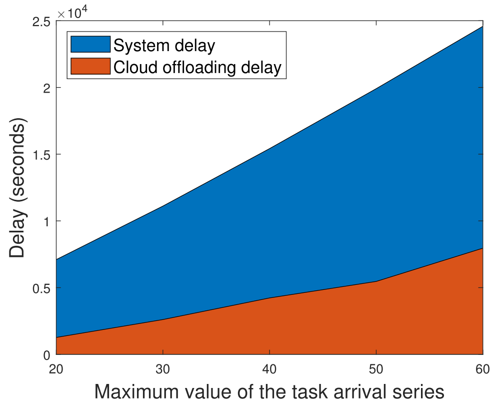


Fig. 20. The proportion of cloud offloading delay in system delay.

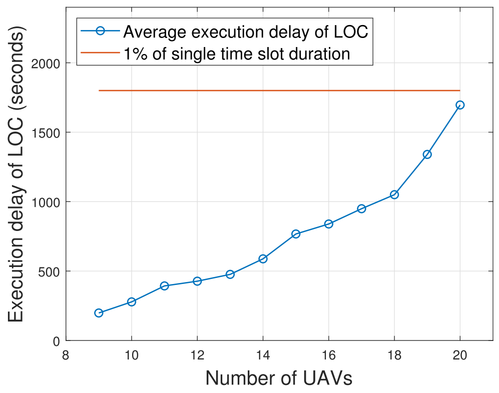


Fig. 21. Execution delay for LOC.

delay of LOC is more approximate to the theoretical lower bound. But the energy consumption is also proportional to V. Therefore, V is the key to achieve the trade-off between delay and energy consumption. When deploying the algorithm in a practical scenario, parameter V should be well designed to adapt the requirement for delay and energy consumption.

Apart from task management capability, whether LOC is able to cope with greater computational pressure has also been studied. Selecting time series of length 50 randomly from dataset A and B, the maximum value of the series is set to $\{20, 30, 40, 50, 60\}$ to realize more tasks for computation. Fig. 19 indicates that with the greater computational pressure, LOC can still achieve better performance than NcC and CoC. The results imply that LOC has the ability to cope with heavy mission's stress. Higher number of tasks results in more computation than multi-UAV system capability. Therefore, more tasks have to be offloaded to the cloud, corresponding to the Fig. 20. The Cloud offloading delay term in Fig. 20 means the delay caused by offloading to the cloud in LOC's decision. However, delay caused by offloading to the cloud does not become the main ingredient of the system delay, which means that LOC still attempts to fully utilize the computation capability of the multi-UAV system for delay minimization.

Finally, the efficiency of the algorithm has been investigated. In all the experiments mentioned above, the execution time required for making a cooperative offloading decision in each slot is counted. Statistics are recorded in Table 4, involving the minimum, first quartile, median, third quartile, maximum and average value for executing LOC. It can be observed that executing LOC requires $167.7 \sim 231.2$ms with the average time 198 ms. The LOC algorithm was executed by single CPU core, and the execution delay is small enough to be negligible compared to the time slot length 3 minutes. 3 minutes is a typical value of task offloading period in edge computing scenarios [35]. Actually, the execution time can be reduced remarkably through some acceleration schemes, e.g., C++ implementation, etc. Furthermore, LOC utilizes one CPU core for an extremely small time slice while the computation tasks are more likely to dominate all cores and all time slices. In other words, rare resource competition between LOC and the actual computation tasks happens in the real world. Thus, executing LOC will not restrict the system performance. Apart from that, additional experiment has been done. As Fig. 21 illustrates, expanding the scale of the UAV cluster, the execution delay of LOC does not exceed 1% of single time slot duration.

In summary, the LOC algorithm is able to make a better trade-off between delay and energy consumption. System delay can be reduced significantly to satisfy the real-time requirements under the energy budget. In addition, LOC has the ability to adjust the workload to avoid the overload of each node, and scenarios' diversity will not restrict its performance. When facing greater computational pressure, LOC still achieves better performance. Its computational efficiency allows LOC to be deployed in real-world multi-UAV edge-cloud systems.

TABLE 4
Execution Delay of LOC

min	25%	median	75%	max	avg
167.7 ms	185.3 ms	193.6 ms	207.0 ms	231.2 ms	198.0 ms

6 Conclusion

This paper has investigated the delay-aware cooperative task offloading problem for the multi-UAV enabled edge-cloud computing system. Specifically, network congestion in UAV-to-UAV communications and the complicated air-to-ground channel are considered to construct the transmission delay model. The feature of parallel computing in distributed system is considered to model the computation delay. Furthermore, the delay-optimal cooperative task offloading problem is formulated and a sub-optimal algorithm is proposed accordingly. Based on Lyapunov optimization and convex approximation, the proposed algorithm has the ability to solve the problem efficiently within limited energy budget. To validate the accuracy of the model, a practical UAV-enabled edge computing platform is constructed. Measurements performed on the platform verify that the model is consistent with reality. Data-driven simulation experiments prove that the proposed algorithm can fully utilize available system energy to satisfy the real-time requirements. The performance of the algorithm is stable when facing different environments and greater computation pressure. Future efforts are worthy doing to investigate the flight trajectory optimization algorithm to coordinate with LOC. Besides, deploying the proposed algorithm in real world thoroughly is also on the agenda.

REFERENCES

[1] G. Zhang, Q. Wu, M. Cui, and R. Zhang, "Securing UAV communications via joint trajectory and power control," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1376–1389, Feb. 2019.

[2] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[3] 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 J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.

[4] K. R. Lakhani and M. Creaner, "Connected drones: A new perspective on the digital economy," 2017. [Online]. Available: https://www.huawei.com/en/technology-insights/industry-insights/outlook/mobile-broadband/xlabs/insights-whitepapers/connected-drones-a-new-perspective-on-the-digital-economy

[5] W. Ma, X. Liu, and L. Mashayekhy, "A strategic game for task offloading among capacitated UAV-mounted cloudlets," in *Proc. IEEE Int. Congr. Internet Things*, 2019, pp. 61–68.

[6] S. Hosseinalipour, A. Rahmati, and H. Dai, "Interference avoidance position planning in dual-hop and multi-hop UAV relay networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7033–7048, Nov. 2020.

[7] I. Valiulahi and C. Masouros, "Multi-UAV deployment for throughput maximization in the presence of co-channel interference," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3605–3618, Mar. 2021.

[8] Y. Zhou et al., "Secure communications for UAV-enabled mobile edge computing systems," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 376–388, Jan. 2020.

[9] 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 Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.

[10] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function requirements in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2672–2685, Nov. 2019.

[11] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021.

[12] Z. M. Fadlullah, D. Takaishi, H. Nishiyama, N. Kato, and R. Miura, "A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks," *IEEE Netw.*, vol. 30, no. 1, pp. 100–105, Jan./Feb. 2016.

[13] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

[14] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3984–3997, Sep. 2020.

[15] 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 J.*, vol. 7, no. 8, pp. 6898–6908, Aug. 2020.

[16] B. Chen, H. Zhou, J. Yao, and H. Guan, "RESERVE: An energy-efficient edge cloud architecture for intelligent multi-UAV," *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 819–832, Mar./Apr. 2022.

[17] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Cellular-connected multi-UAV MEC networks: An online stochastic optimization approach," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6630–6647, Oct. 2022.

[18] J. Almutairi, M. Aldossary, H. A. Alharbi, B. A. Yosuf, and J. M. H. Elmirghani, "Delay-optimal task offloading for UAV-enabled edge-cloud computing systems," *IEEE Access*, vol. 10, pp. 51575–51586, May 2022.

[19] C. Ashraf, "Verizon 5G ultra wideband and edge compute enable smart drones to navigate weather," 2022. [Online]. Available: https://www.verizon.com/about/news/verizon-5g-smart-drones-navigate-weather

[20] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Ind. Inform.*, vol. 16, no. 8, pp. 5505–5516, Aug. 2020.

[21] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[22] L. Zhang and N. Ansari, "Optimizing the operation cost for UAV-aided mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6085–6093, Jun. 2021.

[23] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 46–52, May 2018.

[24] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, and Y. Ren, "Stability of cloud-based UAV systems supporting Big Data acquisition and processing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 866–877, Third Quarter 2019.

[25] H.-P. Shiang and M. van der Schaar, "Online learning in autonomic multi-hop wireless networks for transmitting mission-critical applications," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 5, pp. 728–741, Jun. 2010.

[26] B. Liu, W. Zhang, W. Chen, H. Huang, and S. Guo, "Online computation offloading and traffic routing for UAV swarms in edge-cloud computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8777–8791, Aug. 2020.

[27] R. Duan, J. Wang, C. Jiang, Y. Ren, and L. Hanzo, "The transmit-energy vs computation-delay trade-off in gateway-selection for heterogenous cloud aided multi-UAV systems," *IEEE Trans. Commun.*, vol. 67, no. 4, pp. 3026–3039, Apr. 2019.

[28] N. Toorchi, F. Hu, S. Pudlewski, E. Bentley, and S. Kumar, "Volcano routing: A multi-pipe high-throughput routing protocol with hole avoidance for multi-beam directional mesh networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2981–2996, Dec. 2020.

[29] O. Esrafilian, R. Gangula, and D. Gesbert, "Autonomous UAV-aided mesh wireless networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 634–640.

[30] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro air vehicle link (MAVLink) in a nutshell: A survey," *IEEE Access*, vol. 7, pp. 87658–87680, 2019.

[31] Q. Luo, C. Li, T. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 2897–2909, Sep./Oct. 2022.

[32] X. He, R. Jin, and H. Dai, "Multi-hop task offloading with on-the-fly computation for multi-UAV remote edge computing," *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 1332–1344, Feb. 2022.

[33] Y. Zhuang et al., "Data collection with accuracy-aware congestion control in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 5, pp. 1068–1082, May 2019.

[34] S. K. Kaul and R. D. Yates, "Timely updates by multiple sources: The M/M/1 queue revisited," in *Proc. 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.

[35] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[36] S. Wang, X. Zhang, Z. Yan, and W. Wenbo, "Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4295–4306, Jun. 2019.

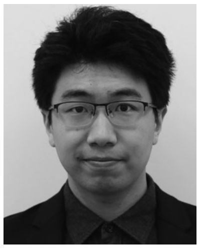
[37] Google, "gRPC," 2018. [Online]. Available: https://grpc.io/

[38] R. A. Light, "Mosquitto: Server and client implementation of the MQTT protocol," *J. Open Source Softw.*, vol. 2, no. 13, 2017, Art. no. 265.

[39] J. Yang, Z. Shan, and Z. Chen, "Research and practice of swoole asynchronous multithreading design method," in *Proc. 4th IEEE Int. Conf. Comput. Commun.*, 2018, pp. 2163–2169.

[40] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4576–4589, Sep. 2019.

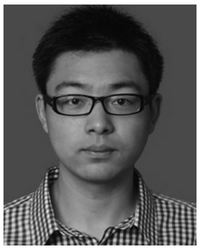
[41] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 328–331, Jun. 2018.
[42] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.
[43] Y. S. Meng, Y. H. Lee, and B. C. Ng, "Empirical near ground path loss modeling in a forest at VHF and UHF bands," *IEEE Trans. Antennas Propag.*, vol. 57, no. 5, pp. 1461–1468, May 2009.
[44] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization versus containerization to support PaaS," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2014, pp. 610–614.
[45] D. Zhang et al., "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 880–893, Apr. 2020.
[46] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[47] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 15, no. 5, pp. 20–53, Dec. 2016. [Online]. Available: https://queue.acm.org/detail.cfm?id=3022184
[48] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," Aug. 2021, *arXiv:2107.08430*.
[49] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Syst.*, vol. 29, no. 2, pp. 143–155, 2012.
[50] K. Cetinski and M. B. Juric, "AME-WPC: Advanced model for efficient workload prediction in the cloud," *J. Netw. Comput. Appl.*, vol. 55, pp. 191–201, Sep. 2015.
[51] J. Xue, F. Yan, A. Riska, and E. Smirni, "Scheduling data analytics work with performance guarantees: Queuing and machine learning models in synergy," *Cluster Comput.*, vol. 19, no. 2, pp. 849–864, Apr. 2016.
[52] F. Kaup, P. Gottschling, and D. Hausheer, "PowerPi: Measuring and modeling the power consumption of the raspberry Pi," in *Proc. 39th Annu. IEEE Conf. Local Comput. Netw.*, 2014, pp. 236–243.
[53] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, 2012, pp. 59–66.



Zhuoyi Bai received the BE degree in electronic information and communications from the Huazhong University of Science and Technology, Wuhan, China, in 2020. He is currently working toward the MS degree with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan. His current research interests include edge/distributed computing and the real-time video transmission.

Yifan Lin received the BE degree in electronic information and communications from the Huazhong University of Science and Technology, Wuhan, China, in 2021. He is currently working toward the MS degree with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan. His current research interests include edge computing and the Internet of Things.

Yang Cao (Member, IEEE) is currently an associate professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. From 2011 to 2013, he was with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, Arizona, as a visiting scholar. His research interests include video transmission and edge/distributed computing. He has coauthored 50 papers on refereed IEEE journals and conferences. He was awarded CHINACOM Best Paper Award, in 2010 and awarded Microsoft Research Fellowship, in 2011.



Wei Wang (Senior Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He is currently a professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. His research interests include PHY/MAC design and mobile computing in wireless systems. He served on TPC of INFOCOM and GLOBECOM. He served as editors for *International Journal of Communication Systems*, *China Communications*, and guest editors for *Wireless Communications and Mobile Computing* and the *IEEE COMSOC MMTC Communications*.