2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovations

Joint Computation Offloading and Routing Optimization for UAV-Edge-Cloud Computing Environments

Baichuan Liu*, Huawei Huang†, Song Guo‡, Wuhui Chen*, Zibin Zheng*
*School of Data and Computer Science, Sun Yat-sen University, China
National Engineering Research Center of Digital Life, Sun Yat-sen University Guangzhou 510006, China
†Academic Center for Computing and Media Studies, Kyoto University, Japan
‡Department of Computing, The Hong Kong Polytechnic University, Hong Kong

lbaichuan@126.com, hwhuang@media.kyoto-u.ac.jp, song.guo@polyu.edu.hk, {chenwuh, zhzibin}@mail.sysu.edu.cn

***Abstract*—Computation offloading is significant to the UAV swarms by migrating computational tasks from the UAV swarms to the edge or cloud computing infrastructure. However, when studying the computation offloading problem, existing approaches do not fully consider the characteristics of UAV swarms and do not differentiate the characteristics of cloud computing and edge computing well. In this paper, we study a joint computation offloading and routing optimization problem for UAV swarms under an UAV-Edge-Cloud computing architecture. Our contributions can be summarized as the following three aspects. First, to fully optimize the characteristics of UAV swarms, we jointly consider the computation offloading and routing for UAV swarms. Then, to highlight the characteristics of cloud computing and edge computing, we propose a novel three-layer computing model joint computation offloading and routing problem. Finally, we design a polynomial near-optimal approximation algorithm to solve the joint optimization problem using the Markov approximation technique. Finally, our simulation results demonstrate the high efficiency of our proposed algorithm.**

***Index Terms*—UAV, Computation Offloading, Edge Computing, Cloud Computing**

I. Introduction

Unmanned Aerial Vehicle (UAV) swarms have been widely used for service provisioning in many different scenarios. For examples, the intelligent distance measurement based on UAV image mosaic technology can be applied to road traffic accident scene [1]. Geographers can apply UAVs for mapping [2]. The UAV technology is also applied to disaster monitoring [3]. All of these work mentioned above have the following common characteristics: 1) the computation offloading were applied towards a high efficiency; 2) the topological structure of the UAV swarm is relatively stable compared with the scenarios of package delivery [4], detection and fighting against the wildfire [5]. When UAVs executes tasks like shooting and mapping, a large amount of streaming data is generated. These tasks form a workflow with the characteristics of big data and time-sensitive. Improving the QoS (Quality of Service) of UAV system is important [6]. Many works have been proposed in the literature to address the streaming workflow allocation problem [7] [8]. However, due to limited onboard resources, UAVs cannot support resource-intensive applications. Fortunately, the edge computing can provide low-latency and energy-efficiency services while cloud computing provides powerful sufficient resources. Therefore, offloading computation tasks from UAV swarms to an edge or a cloud can help improve the efficiency of UAVs. In this paper, we study the computing offloading under a stable topology in the UAV-Edge-cloud environment.

Corresponding author: Wuhui Chen(chenwuh@mail.sysu.edu.cn)

There are many related studies trying to solve the computation offloading problem. Some try to optimize the deployment of application [9] [10], while some others paying attention to the cloud computing [11]. For example, Kovachev et al. [12] studied the computation offloading from mobile devices into the cloud. Namazkar et al. [13] put forward a dynamic approach for energy optimization of cloud computing. We also find some works focus on the edge computing. For example, Chen et al. [14] adopted a game theoretic approach for achieving efficient computation offloading in a distributed manner for Mobile-Edge Cloud Computing. Zhang et al. [15] explored the joint optimization on computation offloading and resource allocation of edge computing. From the related work, we have the following findings.

1) Existing approaches do not fully consider the characteristics of UAV swarms. In current approaches, routing problem has been ignored when studying the computation offloading problem. However, UAV swarms generate large amount of data that require collaborative communication for transmission. Thus the routing strategy would greatly affect the overall performance of the whole UAV swarm system. Therefore, it is significant to jointly optimize the computation offloading and routing problem for UAV swarms.
2) Existing approaches do not differentiate the characteristics of cloud computing and edge computing well when studying the computation offloading problem. In current approaches, there are strong assumptions that cloud computing can provide services satisfying the latency constraints and edge computing itself can provide powerful enough resources for UAV swarms. However, in reality, when large-scale data is under transmission between the UAV swarms and the remote cloud, the induced latency and energy-consumption may fail to satisfy the strict latency requirement of UAV swarms. Besides, edge

computing can provide services for UAV swarms with low latency and energy-efficiency, but its computational capacity is limited. Therefore, it is important to study the computation offloading and routing problem in UAV-Edge-Cloud hybrid computing architecture.

Motivated by these findings, in this paper, we study a joint computation offloading and routing optimization for UAV swarms in UAV-Edge-Cloud computing architecture shown in Fig.1(a). Our efforts can be identified by the following three points. First, to fully consider the characteristics of UAV swarms, we **J**ointly optimize the **C**omputation **O**ffloading and **R**outing **(JCOR)** for UAV swarms. Second, to differentiate the characteristics of cloud computing and edge computing, we propose a novel three-layer computing model. Third, based on it, we then formulate a joint computation offloading and routing problem. The major contributions of this paper can be summarized as:

1) We propose a novel UAV-Edge-Cloud computing architecture combining the advantages of both edge computing and cloud computing, ensuring the low transmission delay and high computing capability.
2) Based on the three-layer computing model, we formulate the **JCOR** problem.
3) Using the Markov approximation technique [16], we then design a polynomial near-optimal approximation algorithm to solve the **JCOR** problem.
4) The simulation results show that our approach is feasible and outperforms the widely adopted shortest path first algorithm.

The rest of this paper is organized as follows. Section II gives an overview of existing related work. Section III introduces the system model and formulation the **JCOR** problem. Section IV presents the Markov Approximation based algorithm. Experiment and simulation results are shown in section V. Section VI concludes this paper.

II. Related Work

A. Communication Issues of UAV Swarms

Existing studies investigating UAV communications can be mainly classified into three categories [17]:

1) leveraging UAVs to improve network connectivity;
2) enhancing information collection ability through UAVs;
3) and intra-networking issues for a swarm of UAVs.

To improve network connectivity, multi-hop relays between aerial and ground networks have been realized by adopting UAVs. For example, Goodemeier et al. [18] proposed to switch the roles of UAVs (such as scout node, relay, articulation point and returnee node) based on current network status, thus to facilitate self-optimized Air-to-Ground connectivity.

For information collecting enhancement, UAVs are used to rapidly and autonomously acquire network information. For example, Motlagh et al. [19] exploited UAVs to monitor traffic conditions and pedestrians for eHealth applications.

For UAV swarm networks, Mobile Ad Hoc Network (MANET) architecture can be integrated into UAV communications to construct Flying Ad Hoc Networks (FANET). For example, Gupta et al. [20] explored the dynamic 3D topology of FANET to seek solutions to routing and connecting. Sharma et al. [21] proposed a driver behavior detection approach, which is based on multi-UAVs coordinated VANETs.

B. UAV-Swarm based Networking Architectures

An increasing number of articles leveraging the UAV-aided vehicular networks have emerged in recent years. For example, Zhou et al. [22] built an aerial-ground cooperative vehicular networking architecture and validated its performance through road tests. In this architecture, UAVs form an aerial sub-network to aid the ground vehicular sub-network through air-to-air (A2A) and air-to-ground communications. The aerial sub-network not only collects network and road information for vehicles but also performs as an intermediate relay in V2V communications. Oubbati et al. [23] proposed an UAV-Assisted Vehicular Routing Protocol (UAVR) for vehicular networks. UAVs are exploited to monitor traffic density and vehicle connectivity status, and then exchange the collected information with vehicles through dedicated messages. Based on that information, UAVR can guide vehicles to find the best multi-hop path for V2V communications. To fully leverage the potential of UAVs in enhancing the performance of vehicular networks, Shi et al. [17] designed a new Drones Assisted Vehicular Network (DAVN) architecture, which integrates UAVs, vehicles and infrastructures simultaneously.

C. Position of this Paper

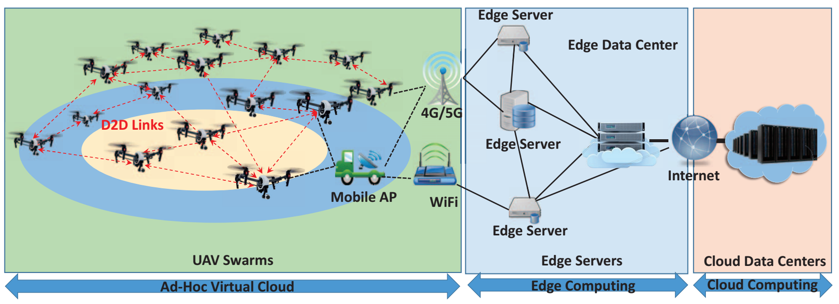
Through the literature review, we found that very few twork has emphasized on the collaborating coordination under the UAV based air-ground collaborating networks. To fill this gap, we study the joint optimization towards the computation offloading and routing problem for UAV swarm based collaborating networks in this paper.

III. System Model and Problem Formulation

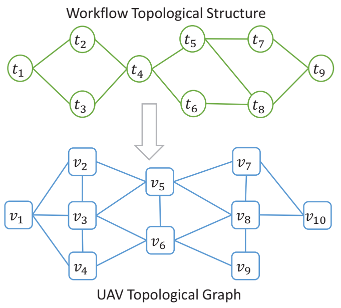
A. System Model

Under the three-layer architecture shown in Fig. 1(a), we consider a UAV-Edge-Cloud computing environment with a stable topological structure in a specific computing window, in which no UAVs and servers join or leave and all connection links keep stabilized. A workflow consists of several tasks, and streaming data is constantly being transferred between tasks. A group of UAV connects with each other via wireless connections within a swarm. To execute the workflow, all its component tasks should be allocated to an UAV swarm, in which routing paths need to be found for streaming data between tasks just shown as Fig.1(b). Furthermore, we consider an UAV-Edge-Cloud computing architecture with powerful edge and cloud servers which have a higher delay on networks but can afford more compute capability.

We use $\mathbb{W} = \{w_1, w_2, ..., w_n\}$ to denote a set of workflow. Each workflow $w \in \mathbb{W}$ contains a set of tasks which can be represented as $\mathbb{T}_w$. A task can be described as $t = \{suc_t, o_t\}$, where suc_t denotes the set of successor tasks of task t and o_t denotes the needed processing resource of task t. The



(a) Illustration of UAV-Edge-Cloud computing architecture.


Workflow Topological Structure
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}
UAV Topological Graph

(b) Illustration of allocating tasks in a workflow to UAV swarm.

Fig. 1: System Model Illustration

bandwidth needed in the data link from task t to task t' can be described as $l_{t,t'}$. We use task segment $\langle t, t' \rangle$ to denote the two adjacent tasks in a workflow, which indicates $t' \in suc_t$.

We consider an UAV–Edge–Cloud computing model with a topological structure $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ denotes the set of nodes that can be UAVs, edge servers or cloud servers. $\mathbb{E} = \{e_{v,v'} | v, v' \in \mathbb{V}\}$ is the link set. We use $e_{v,v'}$ to denote the one-hop link between nodes v and v'. In particular, link delays in different networks are different. For example, communication delay in an UAV network is very low because it is a LAN. To the contrast, link delays from an UAV to an edge server and from an edge server to a cloud server can be terrible. We then define $d_{v,v'}$ to indicate the delay factor of one-hop link $e_{v,v'}$. For convenience, we assume that all the edges in the graph are undirected. We denote c_v the processing rate of the node v. Furthermore, for each link $e_{v,v'} \in \mathbb{E}$, we let $b_{v,v'}$ denote its bandwidth. For any two nodes v and v', $p_{v,v'}$ denotes the candidate path set from node v to node v'. For each $p \in P_{v,v'}$ it can be described as $p = \{e_{v,v_1}, e_{v_1,v_2}, ..., e_{v_n,v'}\}$. The major notations used in this paper are listed in Table 1.

TABLE I: Notations and description

Notations	Description
$\mathbb{W}$	Set of all workflows
$\mathbb{T}_w$	Set of all tasks in a workflow $w \in \mathbb{W}$
suc_t	Set of successor tasks of task t
o_t	Needed processing resource of task t
$l_{t,t'}$	Bandwidth needed in the data link from task t to task $t' \in suc_t$
$\langle t, t' \rangle$	Task segment including two adjacent tasks in a workflow
$\mathbb{V}, \mathbb{E}$	Network topology with node set $\mathbb{V}$ and link set $\mathbb{E}$
$e_{v,v'}$	One-hop link between node v and v'
$d_{v,v'}$	Delay factor of one-hop link $e_{v,v'}$
c_v	Processing rate of the node v
$b_{v,v'}$	Bandwidth of link $e_{v,v'}$
$x^w_{t,v}$	Binary variable indicating whether task $t \in w$ is allocated to node v
$y^{w,p}_{t,t'}$	Binary variable indicating whether a task-segment $\langle t, t' \rangle$ select candidate path p to transfer data
ξ_w	Binary variable indicating whether a workflow $w \in \mathbb{W}$ is satisfied
Φ	The throughput of all satisfied workflows assigned to UAV swarm
Δ	The computation cost of all tasks
Ω	The routing cost of all tasks

B. The *JCOR* problem

1) Definition of variables: First of all, we define $x^w_{t,k}$ as a binary variable indicating whether task t is allocated to an UAV or server v, where $t \in w$, $w \in \mathbb{W}$ and $v \in \mathbb{V}$:

$$x^w_{t,k} = \begin{cases} 1, & \text{if } t \in w \text{ is allocated to node } v \in \mathbb{V}, \\ 0, & \text{otherwise} \end{cases}$$

Because every task segment $\langle t, t' \rangle$ needs a routing path, we then define another binary variable $y^{w,p}_{t,t'}$ to denote whether tasks t and t' allocated to node v and v', respectively, and select candidate path $p \in P_{v,v'}$ to transfer data:

$$y^{w,p}_{t,t'} = \begin{cases} 1, & \text{if task } t \text{ and } t' \text{ in the task segment } \langle t, t' \rangle \text{ select} \\ & \text{the candidate path } p \in P \text{ as a routing path,} \\ 0, & \text{otherwise} \end{cases}$$

We also define a binary variable ξ_w to indicate whether workflow $w \in \mathbb{W}$ is satisfied with a feasible configuration. If all the task segment $\langle t, t' \rangle$ in a workflow $w \in \mathbb{W}$ is satisfied with a feasible configuration. Its meaning is explained as follows:

$$\xi_w = \begin{cases} 1, & \text{all the task segments } \langle t, t' \rangle \text{ in a workflow } w \in \mathbb{W} \\ & \text{can be assigned to an UAV swarm and find a} \\ & \text{routing paths for each,} \\ 0, & \text{otherwise} \end{cases}$$

2) Constraints: We first claim that each task can be only assigned to at most one UAV or server at the same time. We have:

$$\sum_{t \in w} x^w_{t,v} \leq 1, \forall w \in \mathbb{W}, \forall v \in \mathbb{V}. \tag{1}$$

If a workflow is satisfied with a feasible configuration, we have:

$$\sum_{t \in w} \sum_{v \in \mathbb{V}} x^w_{t,v} = \xi_w \cdot |w|, \forall w \in \mathbb{W}, \tag{2}$$

where $|w|$ indicates the number of tasks in workflow w. It means that if a workflow w is satisfied, i.e., $\xi_w = 1$, the total number of UAVs and servers assigned to execute this workflow should be equal to the number of tasks in this workflow.

Then, we consider the data transmission between two tasks in a task segment $\langle t, t' \rangle$ with the goal to find a complete end-to-end routing path for the UAVs or servers executing the tasks. We describe this constraint using the following equation:

$$\sum_{t \in w, t' \in w} \sum_{p \in P_{v,v'}} y_{t,t'}^{w,p} = x_{t,v}^{w} \cdot x_{t',v}^{w}, \forall w \in \mathbb{W}, \forall t' \in suc_t. \quad (3)$$

In (3), $x_{t,v}^{w} \cdot x_{t',v}^{w} = 1$ indicates that the task t and t' are confirmed to be allocated to node v and v', respectively. Thus a routing path for this pair of tasks is selected from the given candidate path set $P_{v,v'}$ to transfer streaming data.

In practice, the aggregated traffic rate on each link $e_{v,v'} \in \mathbb{E}$ can not exceed its bandwidth. We describe this constraint as follow:

$$\sum_{w \in \mathbb{W}} \sum_{t \in w, t' \in w} l_{t,t'} \leq b_{v,v'}, y_{t,t'}^{w,p} = 1. \quad (4)$$

C. Problem Formulation

To measure the performance under a configuration of workflow assignment and routing paths in the UAV–Edge–Cloud computing model, we mainly consider three objective perspectives, i.e.,throughput, computation cost and routing cost.

First, we define a way to measure the throughput of a network configuration. Throughput is a metric that measures the number of satisfied workflow running in an UAV–Edge–Cloud computing model. If one of the task segments in the workflow can not find a satisfied path, then the whole workflow cannot run normally. In this situation, it does not make sense even though the rest of tasks in the workflow are executed as normal. To calculate the throughput of a configuration, we only consider the satisfied workflows. In detail, we measure the throughput in the following way:

$$\Phi = \sum_{w \in \mathbb{W}} \sum_{t \in w} \sum_{t' \in suc_t} l_{t,t'} \cdot \xi_w. \quad (5)$$

According to (5), Φ is closely related to the number of satisfied workflows.

Next, we measure the computation cost which significantly reflects the effect to the assignment of the tasks. For convenience, we do not directly restrict the number of tasks can be allocated to an UAV. However, we take the computation cost into the measurement to can make restriction to the workload for an UAV. Without loss of generality, the more tasks are allocated to an UAV, the larger computation cost is while the processing rate of a node is limited. Referring to [24], we define the computation cost as follows:

$$\Delta = \sum_{v \in \mathbb{V}} \sum_{w \in \mathbb{W}} \sum_{t \in w} \frac{x_{t,v}^{w} \cdot o_t}{c_v}. \quad (6)$$

In the contrast to throughput, we should lower the computation cost as much as possible. A task assignment with a low computation cost means the allocation is reasonable and all the allocated tasks can be processed evenly.

Then, we measure the routing cost. To conduct the traffic engineering in the networks, the forwarding table space is the critical resource. Thus, the consumption of forwarding table space should be considered as an indicator to the performance. When a routing path is found for a task segment, the routing cost is naturally assumed proportional to the number of traversed nodes along the selected routing path [25]. With respect to the delay in different links, we take the delay factor into consideration. Therefore, we can calculate the routing cost as follows:

$$\Omega = \sum_{w \in \mathbb{W}} \sum_{t \in w} \sum_{t' \in suc_t} \sum_{p \in P_{v,v'}} \sum_{e_{v_i,v_j} \in p} y_{t,t'}^{w,p} \cdot d_{v_i,v_j}. \quad (7)$$

It can be seen that, Ω represents all the delay of one-hop links in the selected routing paths.

Normally, we hope to maximize the number of satisfied workflows which is reflected by throughput, so that more tasks can be executed at the same time and improve the efficiency. On the other hand, the routing cost and computation cost should be as small as possible. Finally, by referring to [25], we formulate the **JCOR** problem as the following cost-efficient utility maximization problem, such that the joint performance P associated with admitted throughput and the two aforementioned cost:

$$\textbf{JCOR} : \max P = \Phi - a\Delta - b\Omega, \quad (8)$$

$$\text{s.t. (1), (2), (3), and (4),}$$

where the a and b in (8) represent the weights of the computation cost and routing cost, respectively. These weight parameters can be tuned to place particular emphasis on different demands.

IV. Algorithm Design

A. Markov Chain Design

We use f_{XY}(shorten as f) to denote a feasible configuration of **JCOR** problem, then we have $f \triangleq \left\{x_{t,v}^{w}, y_{t,t'}^{w,p}, \forall w \in \mathbb{W}, \forall t \in w, \forall t' \in suc_t, \forall v \in \mathbb{V}, \forall p \in P_{v,v'}\right\}$. We use $\mathcal{F}$ to denote the set of all feasible configurations and P_f to denote the system performance under configuration f. To solve the optimization problem (8), we utilize the Log-Sum-Exp Approximation [26] and let each configuration associate with a probability p_f, which indicates the portion time that the configuration f is in use. We then let p_f^* to indicate the optimal probability solution for configuration $f \in \mathcal{F}$. According to the approximation framework proposed in [16], we have:

$$p_f^* = \frac{\exp(\beta P_f)}{\sum_{f' \in \mathcal{F}} \exp(\beta P_{f'})}, \forall f \in \mathcal{F}, \quad (9)$$

where β is a positive constant and related to approximation performance. The optimality gap of Log–Sum-Exp Approximation can be restricted by $\frac{1}{\beta} \log |\mathcal{F}|$ [26]. This inspires us to select a large β to lower the approximation gap.

To construct a time-reversible Markov sequential chain [27] with the stationary distribution p_f^*, we consider each feasible configuration $f_{XY} \in \mathcal{F}$ as a state in Markov sequential chain. Transitions between two states are triggered by mapping a task to a new node or allocating a new routing path for a task segment. Note that mapping a task to a new UAV or server plays a more dominated role than allocating new path for a

Algorithm 1 Routing path finding for unsatisfied $\langle t, t' \rangle$

```
1: for $\forall w \in \mathbb{W}$ do
2:     if $\xi_w == 0$ then
3:         for $\forall \langle t, t' \rangle \in w$ do
4:             randomly select a path $p$ from $P_{v,v'}$,
5:             if the unused bandwidth of $p$ is larger than $l_{t,t'}$ then
6:                 $x^w_{t,v} \leftarrow 1, x^w_{t',v'} \leftarrow 1, y^{w,p}_{t,t'} \leftarrow 1$
7:             end if
8:         end for
9:     end if
10: end for
```

task segment. For convenience, we only consider the condition of states transition caused by mapping one task to a new UAV or server at a time. Thus, when a task mapping appears under a given configuration $f_{XY} \in \mathcal{F}$, we say that the configuration transits from $f_{XY} \in \mathcal{F}$ to $f_{X'Y'} \in \mathcal{F}$ with a nonnegative transition rate $q_{ff'}$. For each transition, to ensure that a time-reversible Markov sequential is satisfied, the following two conditions must be ensured: 1) any two states are reachable from each other, and 2) the detailed balance equation [16] is satisfied, i.e., for all f and f' in $\mathcal{F}$ and $f \neq f'$, it should satisfy:

$$p^*_f q_{f,f'} = p^*_{f'} q_{f'f}, \forall f, f' \in \mathcal{F}, \tag{10}$$

where f' denotes $f_{X'Y'}$.

Under such two conditions, we know that even any two remote states are reachable through multiple steps of state transition. The key idea is to build links connecting different states. Corresponding to our problem, we only allow the system to transit to a new state by mapping a task to a node at a time.

There are several ways to design transition rate $q_{ff'}$ [16]. In this paper we set $q_{ff'}$ as the following manner by referring to [25]:

$$q_{ff'} = \exp\left(\frac{1}{2}\beta\left(P'_f - P_f\right) - \boldsymbol{\tau}\right), \tag{11}$$

in which $\boldsymbol{\tau}$ is a conditional non-negative constant. Obviously, the transition rate $q_{f,f'}$ will grow if the performance gap between $f_{X'Y'}$ and f_{XY} increases if $P_{f'} - P_f > 0$. Therefore, the system tends to transfer to a higher system configuration with the transition rate designed in (11).

*B. Implementation of Algorithm to Solve **JCOR** Problem*

We now implement a Markov-approximation based algorithm (MA) [25] [28] [29] [30] to solve the **JCOR** problem. Our algorithm consists of three parts shown as Alg. 1, Alg. 2 and Alg. 3. Note that, Alg. 3 is the main algorithm, before executing Alg. 3, we need to first run Alg. 1 as initialization for all workflows. When the program begins, a dedicated computing thread is created for each workflow, and a monitoring thread is created to listen to the expiration of timers. Then, Alg. 3 as main algorithm executes properly with the support of Alg. 1 and Alg.2.

$Allocation$(Algorithm 1): At the beginning of the program, all the workflows are unsatisfied because none of them have been assigned to the UAVs or servers already. The program first to allocate the workflow tasks to the UAVs or servers

Algorithm 2 Set timer for a task segment $\langle t, t' \rangle$

```
Input: task segment $\langle t, t' \rangle$
Output: $T^w_{\langle t,t' \rangle}$, $\{p_p\}$, $\{p_s\}$
1: $\widehat{v} \leftarrow$ node to execute $t'$ currently;
2: $\widehat{p} \leftarrow$ current in-use path for $\langle t, t' \rangle$;
3: $\sigma^w_{t'} \leftarrow$ new nodes that task $t'$ can adopt;
4: $v' \leftarrow$ new node to execute $t'$;
5: $v \leftarrow$ node to execute $t$;
6: $P_{v,v'} \leftarrow$ all feasible not-in-use candidate paths for $\langle t, t' \rangle$;
7: $\{p_p\} \leftarrow$ set of in-use path for all $\langle t^p, t' \rangle$ and $t^p \neq t$;
8: $\{p_s\} \leftarrow$ set of in-use path for all $\langle t', t^s \rangle$;
9: if $|\sigma^w_{t'}| \geq 1$ then
10:     randomly select $p \in P_{v,v'}$;
11:     for $\forall \langle t^p, t' \rangle \in w$ and $t^p \neq t$ do
12:         $P_{v^p,v'} \leftarrow$ all feasible not-in-use candidate paths for $\langle t^p, t' \rangle$;
13:         if $|P_{v^p,v'}| \geq 1$ then
14:             $\widetilde{p} \leftarrow$ randomly select a feasible not-in-use path from
                $P_{v^p,v'}$;
15:         end if
16:     end for
17:     for $\forall \langle t', t^s \rangle \in w$ do
18:         $P_{v',v^s} \leftarrow$ all feasible not-in-use candidate paths for $\langle t', t^s \rangle$;
19:         if $|P_{v',v^s}| \geq 1$ then
20:             $\bar{p} \leftarrow$ randomly select a feasible not-in-use path from
                $P_{v',v^s}$;
21:         end if
22:     end for
23:     $P_{f_{XY}} \leftarrow P|_{\widehat{v},\widehat{p},\{p_p\},\{p_s\}}$;
24:     $P f_{X'Y'} \leftarrow P|_{\widehat{v} \leftarrow v', \widehat{p} \leftarrow p', \{p_p\} \leftarrow \{\widetilde{p}\}, \{p_s\} \leftarrow \{\bar{p}\}}$;
25:     generate a random exponentially distributed timer $T^w_{\langle t,t' \rangle}$ for
        $\langle t, t' \rangle$ with mean equal to
            $$\frac{1}{|\sigma^w_{t'}|} \exp(\boldsymbol{\tau} - \frac{1}{2}\beta(P_{f_{X',Y'}} - P_{f_{XY}}))   (12)$$
        and begin to count down;
26: end if
```

according to Alg. 1. For each workflow, program allocates its tasks and find feasible paths randomly. Because of the limitation of bandwidth in network, maybe not all of them are able to get an allocation. Some others may still remain in unsatisfied after each allocation.

$SetTimer$ (Algorithm 2): For a task segment $\langle t, t' \rangle$, we denote v and $\widehat{v}$ as the current UAVs or servers to execute task t and t', respectively, and $\widehat{p}$ as the routing path for $\langle t, t' \rangle$. At first, the computing thread of this workflow randomly selects a new UAV or server v' to execute task t'. Then, during the candidate path set from v to v' , algorithm randomly selects one denoted by p. Because task t' may have more than one predecessor tasks and successor tasks, then the computing thread randomly select candidate paths for all the predecessor tasks and successor tasks, which are represented by $\{\widetilde{p}\}$ and $\{\bar{p}\}$, respectively. Next, the current system performance is computed and recorded as $P_{f_{XY}}$. Once the newly selection of nodes and routing paths are adopted, the performance of system under a new configuration is computed and recorded as $P_{f_{X'Y'}}$. Finally, an exponentially distributed random timer whose mean value is equal to (12) is generated independently with the information of selection about nodes and paths, and then begins to count down.

$Main$ (Algorithm 3): At first, the programs execute Alg. 1 to allocate all the workflows to UAV swarm. Then, for

Algorithm 3 Markov-approx based Algorithm to solve **JCOR**

```
1: initially allocate workflows to UAV swarm according to Algorithm 1;
2: for $\forall w \in \mathbb{W}$ do
3:     if $\xi_w == 1$ then
4:         for $\forall \langle t, t' \rangle \in w$ do
5:             SetTimer($\forall \langle t, t' \rangle$) according to Algorithm 2;
6:         end for
7:     end if
8: end for
9: while system is running do
10:    if $T^w_{\langle t,t' \rangle}$ expires then
11:        $x^w_{t,\widehat{v}} \leftarrow 0$, $y^{w,\widehat{p}}_{t,t'} \leftarrow 0$;
12:        $x^w_{t,v'} \leftarrow 1$, $y^{w,p}_{t,t'} \leftarrow 1$;
13:        for $\forall \langle t^p, t' \rangle \in w$ and $t^p \neq t$ do
14:            update path selection according to $\{p_p\}$;
15:        end for
16:        for $\forall \langle t', t^s \rangle \in w$ and $t' \neq t^s$ do
17:            update path selection according to $\{p_s\}$;
18:        end for
19:        SetTimer($\langle t, t' \rangle$) according to Algorithm 2;
20:        send RESET($P_{X',Y'}, w, \langle t, t' \rangle$) signal to program;
21:        select path for unsatisfied task segment according to Algorithm 1;
22:    end if
23:    if a RESET($P_{f'}, \bar{w}, \langle \bar{t}, \bar{t}' \rangle$) signal is received then
24:        $P_{f_{X'Y'}} \leftarrow P_{f'}$;
25:        refresh and start other Timers $T^w_{\langle t,t' \rangle}(\forall w \in \mathbb{W}, \forall t \in w, \forall t' \in suc_t)$ according to (12);
26:    end if
27: end while
```

each workflow, a computing thread is created to set and maintain a timer for its every task segment. After this, a monitoring thread is created to listen to whether there is any timer expires. If an expiration of timer is detected, system will jump to a new configuration by mapping the specific task to a new UAV or server and adopting the new routing paths according to the information recorded in the expired timer. Then, the computing thread sends RESET($P_{X',Y'}, w, \langle t, t' \rangle$) signal to program to notify such new adoption with the updated system performance $P_{f_{X'Y'}}$. When the program receives a RESET($P_{f'}, \bar{w}, \langle \bar{t}, \bar{t}' \rangle$) signal which means task $\bar{t}'$ has been just mapped to a new UAV and server and changed its routing paths so that the new system performance $P_{f'}$ has been yielded. Then, the monitoring thread takes action to refresh the timers $T^w_{\langle t,t' \rangle}(\forall w \in \mathbb{W}, \forall t \in w, \forall t' \in suc_t)$ according to (12) with the updated $P_{f'}$ and begins counting them down. For each unsatisfied task segment, algorithm 1 run again to allocate it to the UAV swarm.

V. Evaluation

A. Experimental Settings

1) Simulator settings: We using a computing model with topological structure which consists 21 UAVs, 3 edge servers and a cloud server. Each UAV can afford a fixed processing rate from 1 to 100 and connect to at least one UAV or edge server through a wireless connection. Edge servers and cloud servers can afford a processing rate of 500 and 10000, respectively. An edge server can have connections with UAVs and cloud servers. Cloud servers can only connect with edge servers. The delay factor of UAV links are set to 1. The delay factor of links from an UAV to an edge server are set to 5. The delay factor of links from an edge server to a cloud server are set to 6. $\boldsymbol{\tau}$ is set to 0.

2) Information about workflows: We construct two kind of workflows: sequential workflows and parallel workflows. In a sequential Workflow, all tasks are linked sequentially and executed one after another. In parallel workflows, a task may have more than one predecessor tasks or successor tasks (the workflow shown in Fig.1(b) is an example). Tasks in different branches can execute in parallel at the same time. We construct a set of sequential workflows which contains 12 sequential workflows having 72 tasks in total. We also construct a set of parallel workflows which contains 10 parallel workflows having 90 tasks in total. Each task in the workflows need a processing resource at the range from 50 to 1000. The bandwidth needed between two tasks is at the range from 1 to 100.

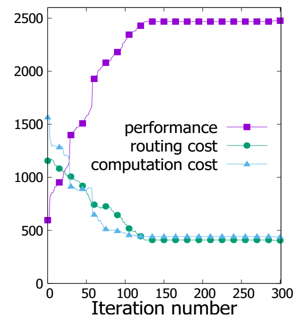
3) Evaluation metrics: We use three evaluation metrics in the experiments: performance, routing cost and computation cost. The performance positively associated with the total throughput of workflows and negatively related to the cost of both routing and computation, is the main metrics in the experiments. Routing cost measure the effectiveness of selecting routing paths. Computation cost measure the effectiveness of mapping tasks to nodes. According to (8), we can obtain the performance by calculating the throughput, routing cost and computation cost.

B. Simulation Results

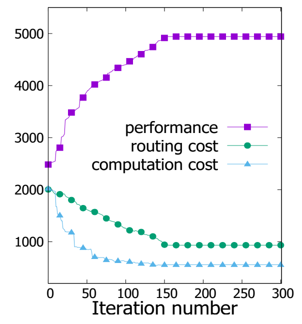
1) Simulation On Sequential Workflows: In this simulation, weight factors a and b are both set to 1. β is set to 10. As shown in Fig.2(a), our algorithm quickly converges to a maximal performance after 130 iterations. Note that the curve representing computation cost plummets several times in Fig.2(a) because servers especially cloud servers can afford a huge computing capability. Therefore once a task needed lots of processing resource is allocated to servers, the overall computation cost will descend obviously.

2) Simulation On Parallel Workflows: Fig.2(b) shows the simulation result on parallel workflows set. Fig.2(b) shows that the performance in this simulation takes more iterations—about 150 iterations, to converge to a maximum, because there is more tasks—90 tasks (compare to the simulation on sequential workflows which includes 72 tasks). In Fig.2(b), the curves that represent routing cost and computation cost show a trend of fluctuating downward. This occurs because with respect to the design of Markov chain, the next configuration is selected randomly, a bad configuration are chosen occasionally while good configurations are chosen with higher probability. For the same reason, the curve that represents performance wavelike rises shown as Fig.2(b).

3) Simulation With Limited Bandwidth: With the objective to execute more workflows at the same time to get a better system performance, our algorithm is designed to dynamically add workflows according to the task assignments and routing

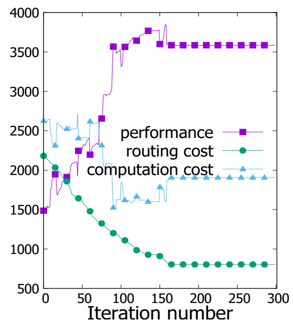


(a) Results of using sequential workflow set.

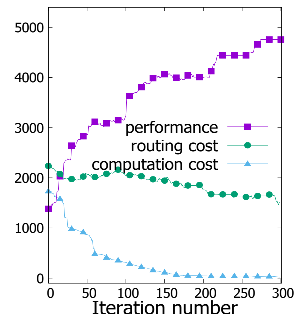


(b) Results of using parallel workflow set.

Fig. 2: Simulation result of different workflow sets.

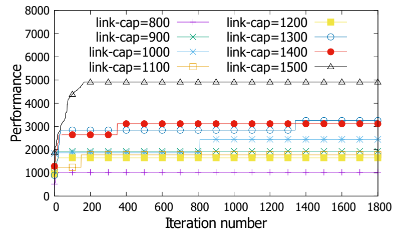


(a) Results under settings a=0, b=1.

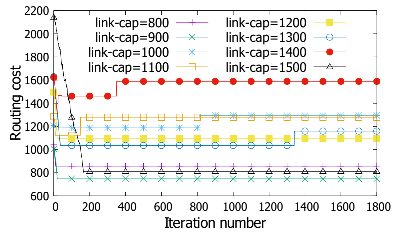


(b) Results under settings a=1, b=0.

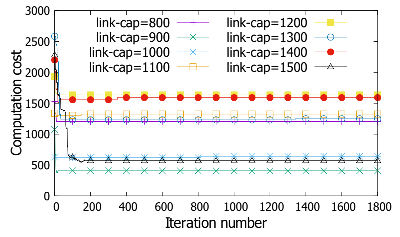
Fig. 3: Simulation result of exploring the effect of weight factors a and b.



(a) System Performance

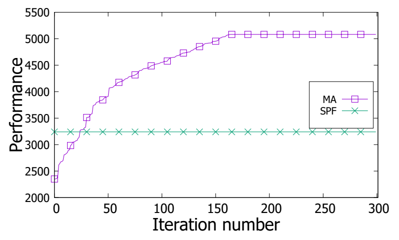


(b) Routing Cost

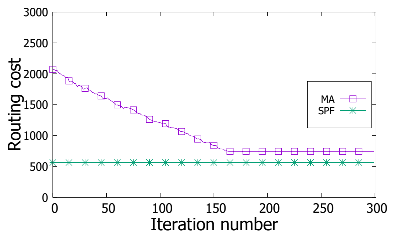


(c) Computation Cost

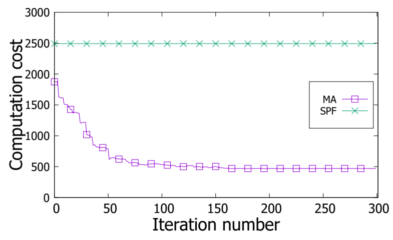
Fig. 4: Results with different link capacities using parallel workflow set, under settings a=1, b=1, β=10.



(a) System Performance



(b) Routing Cost



(c) Computation Cost

Fig. 5: Results of comparing Markov algorithm with Shortest Path First algorithm.

path selections. In this simulation, we lower the bandwidth of all links among the UAVs. We set the bandwidth of UAV connections to 800 and run our algorithm. Then we increase the bandwidth gradually to figure out what is the minimum bandwidth required for the system to execute all workflows. The simulation result is shown in Fig.4. Notes that, although the computation cost and routing cost are not reduced, the performance improve obviously. Because with the increasing of bandwidth, more and more workflows will be satisfied. According to (6) (7) and (8), the minimum computation cost and routing cost will increase and the performance improve as well. We notice that when we set the bandwidth of UAV connections to 1500, we get a simulation result with no significant difference to the one shown in Fig.2(b). So we find out the minimum bandwidth is 1500 approximately.

4) Effect Of weight factor a and b: In different scenarios, people focus on different metrics, which lead to a different assignment of weights to routing cost and computation cost. In this experiment, we explore the effect of weight factor a and b. Firstly, We conduct a simulation using the parallel workflows set. β is set to 10. Weight factors a is set to 0 and b is set to 1, which indicates we ignore the effect of computation cost to system performance. Secondly, we set weight factor a to 1 and b to 0, which indicates we ignore the effect of routing cost to system performance. The results of these two simulations are shown in Fig.3(a) and Fig.3, respectively. Because we only consider a single cost,the curve represents computation cost in Fig.3(a) and the curve represents routing cost in Fig.3 fluctuate randomly while the curve represents routing cost in Fig.3(a) and the curve represents computation cost in Fig.3 are much smoother. Besides, it take more iterations for the MA algorithm to converge to a maximum. What's more, the curves representing performance in Fig.3(a) and Fig.3 reach a smaller maximum compare to the simulation result shown in Fig.2(b). We can find that in this experiment, computation cost plays a more important role than routing cost because the curve represents performance in Fig.3 reach a larger maximum than the one in Fig.3(a).

5) Comparison With Shortest Path First Algorithm: To find out the superiority of joint optimization compared to single optimization, we compare MA algorithm with shortest path first (SPF) algorithm. In this experiment, firstly, we randomly allocate tasks to the nodes and fix their assignment. Then, we find a shortest routing path in the UAV network for all task segments. Secondly, we conduct a simulation with the same setting using MA Algorithm and select several results at different iteration numbers as comparison. The comparison results of two algorithm are shown in Fig.5. When running SPF algorithm, the allocations of task and routing paths are fixed once finishing the initialization, so unlike the MA algorithm, the three evaluation metrics will not change. Because the SPF algorithm get a minimum of routing cost at first, it get a better performance at the beginning. However, SPF algorithm does not optimize the task assignments. As a result, the final computation cost of MA algorithm is much smaller than the SPF's. Therefore, the final performance of MA algorithm is much better than the SPF's.

VI. Conclusion and Future Work

In this paper, we studied the joint computation offloading and routing optimization problem in UAV-edge-cloud computing model. To solve the **JCOR** problem in UAV swarms with object to maximize the throughout and minimize the routing and computation costs at the same time, we proposed an efficient algorithm based on the Markov approximation techniques. The evaluation results demonstrated that our approach is effective in different scenarios. What's more, the simulation results showed better performance compared with a widely adopted shortest path first algorithm. In the future work, we will improve our model with respect to more features of UAV based system such as energy-awareness.

Acknowledgement

This work was supported by the National Key Research and Development Plan(2018YFB1003800), the National Natural Science Foundation of China (61722214), the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme 2016 and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (2016ZT06D211).

References

[1] Y. Liu, B. Bai, and C. Zhang, "Uav image mosaic for road traffic accident scene," in *Automation (YAC), 2017 32nd Youth Academic Annual Conference of Chinese Association of*. IEEE, 2017, pp. 1048–1052.

[2] P. Razi, J. T. S. Sumantyo, D. Perissin, H. Kuze, M. Y. Chua, and G. F. Panggabean, "3d land mapping and land deformation monitoring using persistent scatterer interferometry (psi) alos palsar: Validated by geodetic gps and uav," *IEEE ACCESS*, vol. 6, pp. 12 395–12 404, 2018.

[3] W. Zhang, H. Yu, Z. Yan, and J. Xu, "Study on disaster monitoring technology of mountain fire based on uav transmission line inspection," in *Unmanned Systems (ICUS), 2017 IEEE International Conference on*. IEEE, 2017, pp. 400–403.

[4] M. R. Brust, G. Danoy, P. Bouvry, D. Gashi, H. Pathak, and M. P. Gonçalves, "Defending against intrusion of malicious uavs with networked uav defense swarms," in *Local Computer Networks Workshops (LCN Workshops), 2017 IEEE 42nd Conference on*. IEEE, 2017, pp. 103–111.

[5] C. Phan and H. H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting," in *System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference-7th International Conference on*. IEEE, 2008, pp. 494–498.

[6] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, and K. Nakamura, "Qos-aware robotic streaming workflow allocation in cloud robotics systems," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[7] W. Chen, I. Paik, and P. C. K. Hung, "Transformation-based streaming workflow allocation on geo-distributed datacenters for streaming big data processing," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[8] W. Chen, I. Paik, and Z. Li, "Cost-aware streaming workflow allocation on geo-distributed data centers," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 256–271, Feb 2017.

[9] P. Fan, Z. Chen, J. Wang, Z. Zheng, and M. R. Lyu, "Topology-aware deployment of scientific applications in cloud computing," in *2012 IEEE Fifth International Conference on Cloud Computing*, June 2012, pp. 319–326.

[10] H. Huang, S. Guo, and K. Wang, "Proactive failover for nfv in edge networks," *IEEE Communications Magazine*, 2018.

[11] H. Huang and S. Guo, "Service provisioning update scheme for mobile application users in a cloudlet network," in *IEEE International Conference on Communications*, 2017.

[12] D. Kovachev, T. Yu, and R. Klamma, "Adaptive computation offloading from mobile devices into the cloud," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*. IEEE, 2012, pp. 784–791.

[13] S. Namazkar and M. Sabaei, "Smart cloud-assisted computation offloading system: A dynamic approach for energy optimization," in *Computer and Knowledge Engineering (ICCKE), 2017 7th International Conference on*. IEEE, 2017, pp. 174–180.

[14] 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, 2016.

[15] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, 2018.

[16] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE transactions on information theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[17] W. Shi, H. Zhou, J. Li, W. Xu, N. Zhang, and X. Shen, "Drone assisted vehicular networks: Architecture, challenges and opportunities," *IEEE Network*, 2018.

[18] N. Goddemeier, K. Daniel, and C. Wietfeld, "Role-based connectivity management with realistic air-to-ground channels for cooperative uavs," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 951–963, 2012.

[19] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[20] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

[21] V. Sharma, H.-C. Chen, and R. Kumar, "Driver behaviour detection and vehicle rating using multi-uav coordinated vehicular networks," *Journal of Computer and System Sciences*, vol. 86, pp. 3–32, 2017.

[22] Y. Zhou, N. Cheng, N. Lu, and X. S. Shen, "Multi-uav-aided networks: aerial-ground cooperative vehicular networking architecture," *ieee vehicular technology magazine*, vol. 10, no. 4, pp. 36–44, 2015.

[23] O. S. Oubbati, A. Lakas, N. Lagraa, and M. B. Yagoubi, "Uvar: An intersection uav-assisted vanet routing protocol," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[24] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2017, pp. 1–9.

[25] H. Huang, S. Guo, J. Wu, and J. Li, "Service chaining for hybrid network function," *IEEE Transactions on Cloud Computing*, 2017.

[26] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[27] F. P. Kelly, *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[28] H. Huang, P. Li, S. Guo, W. Liang, and K. Wang, "Near-optimal deployment of service chains by exploiting correlations between network functions," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[29] H. Huang, S. Guo, W. Liang, K. Li, B. Ye, and W. Zhuang, "Near-optimal routing protection for in-band software-defined heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 11, pp. 2918–2934, 2016.

[30] H. Huang, S. Guo, J. Wu, and J. Li, "Joint middlebox selection and routing for software-defined networking," in *IEEE International Conference on Communications*, 2016.