Reinforcement Learning-based Task Offloading of MEC-assisted UAVs in Precision Agriculture

Zih-Yi Yang, Te-Chuan Chiu, and Jang-Ping Sheu
Department of Computer Science, National Tsing Hua University, Taiwan
Emails: s108062586@m108.nthu.edu.tw, theochiu@cs.nthu.edu.tw and sheujp@cs.nthu.edu.tw

***Abstract*—Recently, mobile edge computing (MEC) assisted unmanned aerial vehicles (UAVs) have brought a revolution to the existing precision agriculture (PA). The target UAVs can execute various PA tasks with different heterogeneous resource requirements on the farm. However, due to the stringent service deadline of PA tasks and the battery limitation of UAVs, one of the promising solutions is to offload those computation tasks to MEC servers jointly. This paper explores the MEC-assisted task offloading problem with multi-UAVs under different deadline constraints in uncertain real-world environments. The diverse requirements of PA tasks, the heterogeneous network status, and the dynamic loading of MEC edge servers make the offloading decision an NP-hard problem. Therefore, we propose a reinforcement learning (RL)-based task offloading approach, BANDIT-SCH, to minimize total MEC system costs to achieve online task dispatching and scheduling in uncertain environments without further global information. The experiment results show that the performance of BANDIT-SCH is approximate to the upper bound strategy, which can foresee all edge servers' detailed status.**

***Index Terms*—Mobile Edge Computing, Reinforcement Learning, UAVs, Precision Agriculture.**

I. Introduction

Recently, severe climate change and global population growth explosively, are threatening the production of crops and food dramatically. Precision agriculture (PA) is an emerging technology combining the Internet of Things (IoT), remote sensing, and automated robotic farming to help farmers improve agricultural production intelligently. Unmanned aerial vehicles (UAVs) play a critical role in PA nowadays. The sensors on the UAV periodically send information to the 5G back-end for analyzing valuable farm output. Similarly, UAVs collect real-time information that provides actionable intelligence to farmers. By taking advantage of low deployment cost and flexible mobility, UAVs can support different PA applications ranging from crop monitoring to spraying [1]. However, most PA tasks are computing-intensive and require real-time decision-making [2], which makes UAVs challenging to meet all diverse requirements with limited heterogeneous resources and energy budgets.

Fortunately, mobile edge computing (MEC) is a well-established framework to support UAVs and PA clients with instantaneous computational services available at the edge network. Since the nearby edge server provides the computation service instead of the traditional remote cloud, the offloading latency of the computation task and network congestion through the core network can be significantly reduced. Many companies have started to deploy MEC systems in the 5G/B5G network to assist farmers in achieving PA tasks in the northwestern U.S. and worldwide [3].

This work was supported by the National Science and Technology Council under Grant 112-2222-E-007-002-MY3, National Tsing Hua University under Grant 112QI018E1, and Qualcomm Technologies, Inc. under Grant SOW NAT-487844.

979-8-3503-1090-0/23/$31.00 © 2023 IEEE

There exists intensive research that focuses on designing task offloading policy for MEC applications [4], [5]. Zhang *et al.* [6] address offloading tasks with delay sensitivities in purely MEC systems. Other related works aim to solve the deadline scheduling problem without adopting a UAV plan [7]–[9]. Recently, some literature studied the offloading problems with MEC-assisted UAVs [10]–[12], which optimizes the flight trajectory or minimizes total energy costs. However, the above researchers neglect the diverse deadline requirements of various PA offloading tasks. In our work, UAVs need to provide many types of PA services with different resource and deadline requirements, especially in uncertain real-world environments. It is a novel and challenging task offloading problem compared with existing research.

Reinforcement Learning (RL) has been regarded as a promising solution for designing task-offloading policies [6]. RL-based approaches gain experience through interaction with the environment without domain-specific knowledge or require high-complexity computation efforts. Also, it is a feasible way to achieve fast adaptation in a dynamic and uncertain environment. Previous works [6], [13], [14] advocate deep RL (DRL)-based strategies to solve the offloading problem by considering energy, migration, security, or in real-world environments. To face the uncertainty caused by some stochastic process or unknown information about the environment, it is hard to characterize some fragment information to make the offloading decision. Besides, the search space and the computing cost for applying the DRL-based technique are not practical with resource-limited servers at the edge compared with those in the cloud. Therefore, we choose Multi-Armed Bandit (MAB) as one of the RL frameworks to provide a feasible solution that embraces the uncertainty over time. Recently, [15] formulates the MAB offloading problem in purely edge systems. Furthermore, [16] makes the offloading decisions under non-stationary network dynamics. However, the above MAB-applied works have not yet considered MEC-assisted UAVs and heterogeneous resource allocation, especially in the precision agriculture scenario.

In this paper, we consider MEC-assisted UAVs to serve various PA tasks with different stringent deadlines and resource requirements in uncertain real-world environments. We first formulate the optimization problem and show its NP-hard property. The objective is to find a task offloading policy to minimize total MEC system costs, which means sacrificing less essential data. Then, we proposed a two-stage offloading scheme called BANDIT-SCH, including an RL-

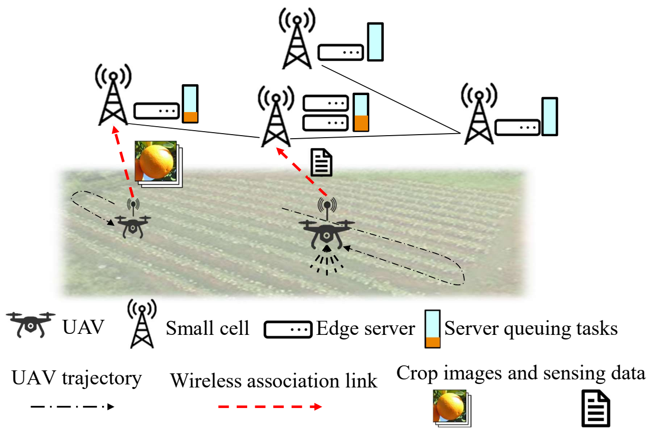


Fig. 1: Scenario of MEC-assisted multi-UAV network in PA.

based Deadline-Aware Bandit Dispatch (DABD) algorithm and a Min-Queuing Scheduling (MQ-SCH) algorithm. BANDIT-SCH enables online task dispatching and scheduling even in uncertain real-world environments without any predetermined model or domain knowledge. We evaluate our approach compared with the state-of-the-art baselines with dynamic loads and stochastic task offloading in MEC systems. The experiment results show that the performance of the BANDIT-SCH is approximate to the upper bound baseline, which can foresee all edge servers' detailed status.

II. System Model and Problem Formulation

A. MEC-assisted UAVs System

The system scenario is shown in Fig. 1. We consider a MEC-assisted multi-UAV system providing various precision agriculture tasks. Total service period $\mathbb{T} = \{1, 2, ..., T\}$ is divided into total T equal time slots. The Base Station (BS) set is denoted as $\mathbb{M} = \{1, 2, ..., M\}$. To provide computing capability closer to users, the MEC edge servers (ES), defined as $\mathbb{K} = \{1, 2, ..., K\}$ with computation capacity f_k, are randomly deployed at the BS. Each ES k maintains a job queue denoted as $\mathbb{Q}_k$ with length $|\mathbb{Q}_k|$.

The UAVs $\mathbb{N} = \{1, 2, ..., N\}$ are randomly deployed on the farm and execute various PA tasks. Specifically, each UAV has a different type of PA mission (e.g., crop image analytics and fertilization) with deadlines and generates tasks at arbitrary times. The task r_n^t generated from UAV n at time t is characterized by a tuple $r_n^t = (s_n^t, \lambda_n, \omega_n)$, where the data size represents s_n^t (bits), λ_n is defined as the computation workload (cycles/bit), ω_n is assigned as the task deadline, respectively. We define a binary variable $o_{n,k}^t \in \{0, 1\}$ as an offloading decision indicator. Each UAV n requires the offloading task to one of the MEC ESs (i.e., $\sum_{k\in\mathbb{K}} o_{n,k}^t = 1$.) Besides, $o_{n,k}^t = 1$ implies that the task was generated from UAV n and further processed at ES k.

B. Task Offloading Model

In our scenario, each BS operates on the same frequency band, which causes mutual inter-cell interference when multi-UAV sends their data to the target BS simultaneously. Therefore, we specify the wireless communication parts in Reference Signal Received Power (RSRP) and Signal-to-Interference-plus-Noise Ratio (SINR) as follows. Each UAV will select the BS with the most strength-received RSRP signal for BS association. We define a binary variable $x_{n,m}^t \in \{0, 1\}$ as an association decision indicator of UAV n associated with BS m at time slot t. The uplink RSRP of UAV n to BS m at time t is given by:

$$rsrp_{n,m}(t) = P - PL_{n,m}(t), \forall n \in \mathbb{N}, \tag{1}$$

where P denotes the constant transmission power of UAVs. $PL_{n,m}(t)$ is obtained by substituting the distance between UAV n and BS m and the dedicated band used by the 5G communication spec [17]. According to the amount of transmitted data of UAVs serving by the same BS, we assume the bandwidth usage of UAV as $b_{n,m}(t)$.

After obtaining the bandwidth usage of UAVs, we derive the interference caused by UAVs under adjacent BSs using the same bandwidth for transmission. The interference of UAV n is calculated with:

$$I_{n,m}(t) = \sum_{m'\in\mathbb{M}\setminus\{m\}} \sum_{n'\in\mathbb{N}\setminus\{n\}} x_{n',m'}^t rsrp_{n',m'}(t) IC_n(\Phi), \\ , \forall n \in \mathbb{N}, \forall m \in \mathbb{M}, \tag{2}$$

where the function $IC_n(\Phi) = \min(\Phi, 1)$ indicates the impact on inter-cell bandwidth used by UAV n, and the proportion of other UAVs using on the same band under neighbor BSs. The $\Phi = \frac{b_{n',m'}(t)}{b_{n,m}(t)}$ represents how the bandwidth $b_{n',m'}(t)$ used by other UAVs in neighboring BSs affects the UAV n bandwidth $b_{n,m}(t)$ ratio. Regarding this, we restrict the ratio range to $\Phi \leq 1$. According to (1) and (2), the SINR of the received signal at the BS m from UAV n is given by:

$$sinr_{n,m}(t) = \frac{rsrp_{n,m}(t)}{I_{n,m}(t) + N_0 B_{n,m}(t)}, \forall n \in \mathbb{N}. \tag{3}$$

where the N_0 refers to the noise spectral density of the normal atmosphere.

Therefore, the transmission rate of UAV n at time slot t can be obtained as:

$$rate_{n,m}(t) = b_{n,m}(t) log_2(1 + sinr_{n,m}(t)), \forall n \in \mathbb{N}. \tag{4}$$

The uplink data transmission delay from UAV n to BS m at time slot t can be calculated as:

$$d_{n,m}^{trans}(t) = \frac{s_n^t}{rate_{n,m}(t)}, \forall n \in \mathbb{N}. \tag{5}$$

Note that we assume that the UAV has no additional storage equipment. If the amount of uploading data s_n^t exceeds the uplink rate $rate_{n,m}(t)$ at time t, UAV n will drop these data (i.e., lose essential data s_n^t). Besides, the computation delay of the task from UAV n executed at ES k is given by:

$$d_{n,k}^{exec}(t) = o_{n,k}^t (\frac{s_n^t \lambda_n}{f_k}), \forall n \in \mathbb{N}. \tag{6}$$

C. Task Queuing Model

To characterize the processing progress of each assigned task in the ES k, we denote the processing time information $p_{n,m,k}^j(t)$ of the j-th process from UAV n at time t as:

$$p_{n,m,k}^j(t) = (\tau, d_{n,m}^{trans}(\tau), d_{n,k}^{exec}(\tau), d_{j,n}^{cpu}(t)), \\ \forall n \in \mathbb{N}, m \in \mathbb{M}, \tag{7}$$

where τ represents the initiation time when the UAV n delivers the task, and $d^{cpu}_{j,n}(t)$ represents the actual execution time of ES k from initial time τ of request to current t. The process will pop out of the $\mathbb{Q}_k$ when the j-th process has been successfully processed. The set of job queue set $\mathbb{Q}_k = \{p^1_{n,m,k}(t), \ldots, p^{|\mathbb{Q}_k|}_{n,m,k}(t)\}$, which the execution order of the process in the queue follow the ascending power of the index $j = 1, \ldots, |\mathbb{Q}_k|$.

Before the ES executes the process, two factors affect the time it starts to be serviced. One is the status of the existing queuing tasks in the ES, and the other is the task transfer time from the UAV to the ES. We denoted the task arrival time of j-th process which generated from UAV n at time τ through BS m transfer to the ES k as:

$$\nu^{\tau}_j = \tau + d^{trans}_{n,m}(\tau). \tag{8}$$

Before calculating the task waiting time of each process $j \in \mathbb{Q}_k$, we need to calculate the remaining processing time $\mathcal{R}_{n,k,j}(t)$ of all previous processes in ES which are calculated as:

$$\begin{aligned}\mathcal{R}_{n,k,j}(t) = \mathcal{R}_{k,(j-1)}(t) + \max(\nu^{\tau}_j - t, 0)\\ + (d^{exec}_{n,k}(\tau) - d^{cpu}_{j,n}(t)),\end{aligned} \tag{9}$$

where the remaining processing time of the 1-th process in the queue is $\mathcal{R}_{n,k,1}(t) = \max(\nu^t_1 - t, 0) + (d^{exec}_{n,k}(\tau) - d^{cpu}_{1,n}(t))$. Therefore, the waiting time of the task is to compare the remaining data transfer time $\nu^{\tau}_k - t$ of the current j-process with the remaining processing time $\mathcal{R}_{n,k,(j-1)}(t)$ of the previous $(j-1)$-th process. We can obtain the waiting time of j-th process from UAV n in the ES k at time t as:

$$d^{wait}_{n,k,j}(t) = \max(\max(\nu^t_j - t, \mathcal{R}_{k,(j-1)}(t)), 0), \forall n \in \mathbb{N}. \tag{10}$$

The total offloading delay of the task generated from UAV n at time t offloaded to ES k is given by:

$$\begin{aligned}D^{total}_{n,m,k}(t) = d^{trans}_{n,m}(t) + d^{wait}_{n,k,j}(t) + d^{exec}_{n,k}(t),\\ \forall n \in \mathbb{N}, m \in \mathbb{M}, k \in \mathbb{K}.\end{aligned} \tag{11}$$

Note that we assume the response time of offloading results back to UAV is relatively short, so we neglect the feedback delay in this paper.

The lack of data collection will affect the chance of misjudgment in PA decision-making. Due to the impact of the random incoming traffic and the uncertain status of the heterogeneous servers, the task may be unable to meet its deadline requirements. We define missing data of UAV n at time t as the system cost, which can be formulated as:

$$\Theta^t_n(D^{total}_{n,m,k}(t)) = \begin{cases} s^t_n, & D^{total}_{n,m,k}(t) > w_n \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

D. Problem Formulation

Based on the above models, we define the task offloading problem with deadline constraint in the heterogeneous MEC system, which is formulated as:

$$\min_{\mathbb{O}} \quad \sum_{t\in\mathbb{T}} \sum_{n\in\mathbb{N}} \sum_{m\in\mathbb{M}} \Theta^t_n(D^{total}_{n,m,k}(t)) \tag{13a}$$

$$\text{s.t.} \quad D^{total}_{n,m,k}(t) \le w_n, \forall n \in \mathbb{N}, \forall m \in \mathbb{M}, \forall k \in \mathbb{K}, \tag{13b}$$

$$\sum_{n\in\mathbb{N}} \sum_{j\in\mathbb{Q}_k} p^t_{k,j,n} \le |\mathbb{Q}_k|, \forall k \in \mathbb{K}, \tag{13c}$$

$$\sum_{m\in\mathbb{M}} x^t_{n,m} \le 1, x^t_{n,m} \in \{0,1\}, \forall n \in \mathbb{N}, \tag{13d}$$

$$\sum_{k\in\mathbb{K}} o^t_{n,k} \le 1, o^t_{n,k} \in \{0,1\}, \forall n \in \mathbb{N}, \tag{13e}$$

$$S^t_n \le rate_{n,m}(t), \forall n \in \mathbb{N}, \forall m \in \mathbb{M}, \forall t \in \mathbb{T}. \tag{13f}$$

In (13a), our goal is to finding a task offloading policy $\mathbb{O} = \{o^t_{n,k}, \forall n \in \mathbb{N}, \forall m \in \mathbb{M}, \forall t \in \mathbb{T}\}$ to minimize the cumulative offloading cost of UAVs subject to (13b) - (13f). The (13b) specifies each PA task should meet the service deadline. The (13c) ensures the assigned tasks will not exceed the server's queue length. The (13d) indicates that each UAV can only associate with a single BS for task transmission. The (13e) implies that each UAV can be served by only one MEC server for task offloading. The (13f) specifies that the task will be dropped when the data rate from BS is unsatisfied. Since the optimization variable $o^t_{n,k} \in \mathbb{O}$ is a binary integer variable, and the other constraints preserve linearity, our target problem (13) is formulated as an integer linear programming (ILP) problem.

III. Reinforcement Learning-based Task Offloading Approach

To provide various PA services in uncertain real-world environments, our focused task offloading problem is *NP-hard* and infeasible to achieve the optimal solution in polynomial time. Therefore, we first state the hardness property of our target problem and then adopt Reinforcement Learning (RL) strategy to seek a feasible solution.

A. NP hardness

Theorem 1. *Task offloading problem is $\mathcal{NP}$-hard.*

Proof: This problem is $\mathcal{NP}$-hard and can be proved by a reduction from *generalized assignment problem* [18]. The proof is omitted due to a lack of space. ■

To deal with challenging task offloading problem, we proposed an RL-based approach, *BANDIT-SCH*, which includes a *Deadline-Aware Bandit Dispatch (DABD)* algorithm and a *Min-Queuing Scheduling (MQ-SCH)* algorithm. At each round, when the agent receives the request from the UAV, the *DABD* algorithm first assigns the task to the target ES k and collects the system cost as reward feedback to estimate the future status of the MEC system. Then, the target ES k decides the service order of the $\mathbb{Q}_k$ by *MQ-SCH* to mitigate the waiting period of the queuing tasks.

B. Deadline-Aware Bandit Dispatch Policy

Based on the state-of-the-art bandit algorithm UCB1 [19], we revise it as *DABD* algorithm to solve the stochastic offloading problem and balance the dilemma between explore and exploit jointly. The bandit framework comprises two elements, the action space $\mathcal{A}$ of the arm and the reward function $Re()$, Algorithm 1: Deadline-Aware Bandit Dispatch (DABD)

```
1:  Initialize:
        The request dispatch counter $c \leftarrow 0$,
        $N(a) \leftarrow 0$, $Q(a) \leftarrow 1$, $\forall a \in \mathcal{A}$.
2:  for $t \leftarrow 1$ to $T$ do
3:      while the MEC system receives $r_n^t$ at time $t$ do
4:          Calculate the estimated empirical cost of each
            arms $a \in \mathcal{A}$ by (17), and select the arm with
            the highest upper value.
5:          Get the reward $Re(r_n^t, a^*)$ with (16).
6:          $Q(a^*) \leftarrow Q(a^*) + \frac{(Re(r_n^t,a^*)-Q(a^*))}{N(a)}$
7:          $N(a^*) \leftarrow N(a^*) + 1$
8:          $c \leftarrow c + 1$
9:      end while
10: end for
```

respectively. We take the number of ESs $|\mathbb{K}|$ as the available arms of the MAB framework. The action space is defined as follows:

$$a \in \mathcal{A} = \{1, 2, ..., |\mathbb{K}|\}. \tag{14}$$

Intuitively, DABD leverages the idea of concentration inequality, which provides bounds on how a random variable deviates from the expected action-value $E[Q(a)]$. When the number of decision-making is large enough, the empirical mean action-value of the arm, denoted as $Q(a)$, will be close to $E[Q(a)]$. DABD aims to find a policy to get the maximum empirical mean $Q(a)$. The reward function will guide the decision-making behavior of the task dispatch agent. Traditionally, the RL algorithms bound their reward into [0, 1] to guarantee convergence in mathematics. Therefore, we need to normalize the system cost of the server a given by,

$$\hat{\Theta}_a^{N(a)} = \frac{\Theta_n^t(D_{n,m,k}^{total}(t))}{\sup_{\forall N(a),a} \hat{\Theta}_a^{N(a)}}, \tag{15}$$

where the $N(a)$ is the number of time choose server a, and the $\sup_{\forall N(a),a} \hat{\Theta}_a^{N(a)}$ records the maximum system cost value after select server a for $N(a)$ times. We consider reward as the opposite of the system cost, given by (15). After selecting server a, the reward can be calculated with the following:

$$Re(r_n^t, a) = \begin{cases} -\hat{\Theta}_a^{N(a)}, & |\mathbb{Y}_a| = 0 \\ -(\frac{\hat{\Theta}_a^{N(a)}}{|\mathbb{Y}_a|}), & \text{otherwise} \end{cases}. \tag{16}$$

where we define the set $\{r_n^t \cap \mathbb{H}_k | \hat{\Theta}_n = 0\}$ to record tasks assigned to ES k that meet the deadline requirements. The RL agent evaluates the confidence value of each arm and selects the highest value as offloading decision a^*, given by:

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}(Q(a) + \sqrt{\frac{2ln(c)}{N(a)}}), \tag{17}$$

where c is dispatch counters and $N(a)$ is the number of selection to server a.

Once we get the offloading server decision a^* given by DABD, we can obtain the task offloading indicator as o_{n,a^*}^t. Following the offloading policy $o_{n,a^*}^t \in \mathbb{O}$, the learning goal of DABD is to minimize the cumulative system cost $\sum_{c=0}^{c} -(Re(r_n^t, a^*))$, representing data loss. The pseudo codes of DABD are given in Alg 1. The learning procedure of DABD starts when receiving the new request r_n^t. In each decision round c, the agent evaluates the confidence value of each arm calculated by (17) and chooses the largest one, which has the highest upper confidence value in line 4.

After the agent assigns the r_n^t to ES a^* and gets a negative reward, the remaining step is to update the average estimate value of the a^* and the number of selections $N(a^*)$ in lines 6-8. By accumulating task dispatching experience, the agent will gradually learn a better offloading policy.

Algorithm 2: Min-Queuing Scheduling (MQ-SCH)

```
Input: The dispatch server $a^*$,
       New assigned request $r_n^t$, the execution status of target
       ES $\mathbb{Q}_{a^*}(t)$.
Output: The execution order $j^\star$ of request $r_n^t$ in $\mathbb{Q}_{a^*}$
1:  Initialize:
        $j^\star \leftarrow -1$, $\hat{\mathbb{Q}_{a^*}} \leftarrow \mathbb{Q}_{a^*}(t)$, $feasible \leftarrow false$,
        $D^{wait} \leftarrow 0$, $D^{wait*} \leftarrow 0$
2:  for $j$ in $|\hat{\mathbb{Q}_{a^*}}|$ down to 1 do
3:      if $j = |\hat{\mathbb{Q}_{a^*}}|$ then
4:          Record the total waiting time of the queuing tasks
            at the tail position, $D^{wait*} \leftarrow \sum_{j \in \hat{\mathbb{Q}_{a^*}}} \mathcal{R}_{n,k,j}(t)$
5:      else
6:          Swap the scheduling position of process from
            $p_{n,m,a}^j$ to $p_{n,m,a}^{j-1}$
7:          Record the total waiting time of the queuing tasks
            at the new position, $D^{wait} \leftarrow \sum_{j \in \mathbb{Q}_{a^*}} \mathcal{R}_{n,k,j}(t)$
8:      end if
9:      if All process $p_{n,m,a}^j \in \hat{\mathbb{Q}_{a^*}}$ meet its deadline then
10:         $feasible \leftarrow$ True
11:         if $D^{wait*} \geq D^{wait}$ then
12:             $D^{wait*} \leftarrow D^{wait}$
13:             $j^\star \leftarrow j$
14:         end if
15:     else
16:         break;
17:     end if
18: end for
19: if $feasible$ is $False$ then
20:     Reject the task $r_n^t$.    ▷ The request not satisfied (13b)
21: else
22:     $\mathbb{Q}_{a^*} \leftarrow \hat{\mathbb{Q}_{a^*}}$    ▷ Update to scheduled queue
23:     return $j^\star$
24: end if
```

C. Min-Queuing Scheduling Mechanism

After the intelligent agent makes the task offloading assignment, the assigned server a^* will schedule the execution order in the job queue $\mathbb{Q}_{a^*}$ to reduce the total waiting time without violating the deadline in (13e). The pseudo codes of MQ-SCH are given in Alg. 2. The main procedure of MQ-SCH is in three steps. First, we push the task in the tail of $\hat{\mathbb{Q}_{a^*}}$ and get the initial cumulative waiting time D^{wait} of the existing tasks in $\hat{\mathbb{Q}_{a^*}}$ in lines 3-5. Second, we check whether the order of the new schedule meets the deadline in lines 2-18. We repeat the above two steps to check the remaining possible position is feasible and make the total D^{wait} less. Last, we confirm whether to reject the task if there has no feasible position in $\hat{\mathbb{Q}}_{a^*}$ in lines 19-24. The scheduling procedure will be interrupted if the swap order of the task does not satisfy the deadline. If there is no feasible position for the task in $\hat{\mathbb{Q}}_{a^*}$, the ES a^* will reject the request r_n^t accordingly.

IV. Performance Evaluation

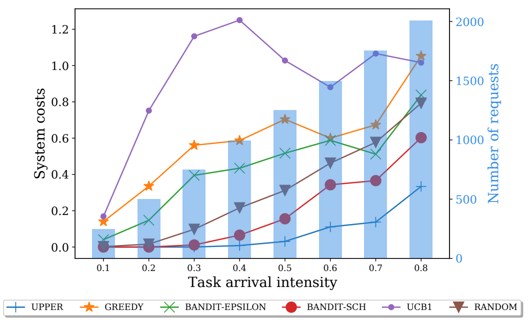
A. Experiment Settings

We evaluate the performance of BANDIT-SCH in a MEC-assisted UAVs network with various realistic network settings. There are 40 BSs on the farm with an area of 1600×700 m^2. We set the transmission radius of the BS as 350 m and operated on the dedicated band with a bandwidth of 100 (MHz). The UAVs perform PA tasks by offloading crop images, which requires data size ranging from 1 MB to 3 MB, and the workload of processing image recognition is 1900 cycles per bit. The deadline ω_n for the PA task is [10, 15] slots. The clock speeds f_k for each heterogeneous server k are set uniformly distributed in [2, 3] (GHz). Besides, we randomly deployed total of five servers, in which the number of CPU cores for each server is set to be uniformly distributed [6, 10]. Therefore, the computation capacity provided by the server is multiplied by the clock speed and number of cores. The maximum queue length of each server $\mathbb{Q}_k$ is uniformly assigned in [3, 8]. Our simulations run on Python 3.6.13 with Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz processor. We evaluate the uncertain MEC environments in that each server has a dynamic load, representing the probability that the server has a random incoming task to serve at an arbitrary time. Each simulation results run averaged over 500 slots and in 20 different episodes. We verify the performance of each offloading scheme with two metrics: *cumulative system cost* and *satisfaction rate*. The cumulative system cost evaluates the service quality of the MEC system. The task offloaded from UAVs may be dropped due to insufficient bandwidth allocated by BS. Besides, we define the satisfaction rate of the server as the number of tasks that meet the deadline divided by the number of tasks successfully received by the server.

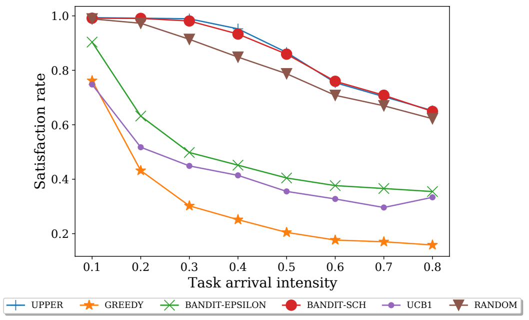
The proposed BANDIT-SCH is compared with five candidate algorithms as follows: *UPPER*, which can foresee all edge servers' detailed status and select the ES with the shortest accumulated waiting time for queuing tasks. Generally, the queuing status of ES is hard to access, so we take the UPPER as the performance upper bound. *GREEDY*, as a heuristic algorithm, the offloading agent assigns the tasks to the ES with the least cumulative system cost. *UCB1*, as a classic MAB algorithm [19], which considers whether the task meets the deadline and gives a binary reward. The binary reward in UCB1 will receive one if the task meets the deadline constraint, or it will receive 0. *BANDIT-EPSILON*, as one of the MAB algorithms with ϵ probability to explore and $(1-\epsilon)$ possibility to exploit the arms to get the minimal cumulative system cost. Finally, *RANDOM*, as the performance lower bound, selects one of the ES randomly.

B. Impact of the Stochastic Offloading and Dynamic Load

We evaluate the performance of BANDIT-SCH and other baselines in two different PA task request patterns from UAVs and under dynamic load in the edge servers. One of the task-generating patterns varies in the task arrival intensity of the system and represents the probability that the UAV sends a request to the MEC system at each time. The enormous

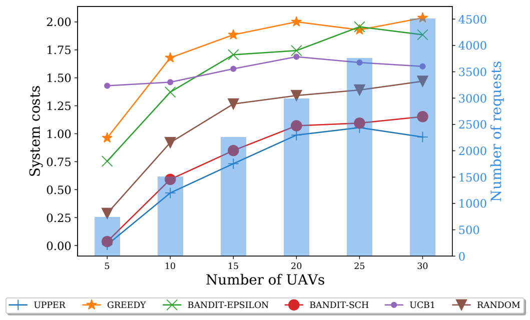


(a) Cumulative system cost.

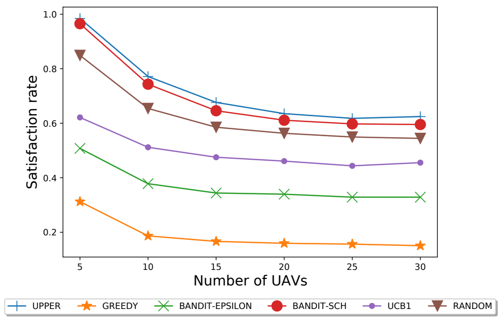


(b) Satisfaction rate.

Fig. 2: The impact on the task arrival intensity.



(a) Cumulative system cost.



(b) Satisfaction rate.

Fig. 3: The impact on the heterogeneous MEC system.

task arrival intensity makes the system more stochastic and challenging to support PA services. In Fig. 2, we evaluate the impact of task arrival intensity from intensity 10% to 80% with a total of five UAVs in the system. Fig. 2(a) presents the cumulative system cost, and Fig. 2(b) shows the satisfaction rate, respectively. As expected, the proposed BANDIT-SCH scheme can find the proper offloading policy and lose less essential data to approximate the UPPER in both performance indexes compared to other baselines under different intensities. The custom reward lets BANDIT-SCH control well in exploration and exploitation to adapt to the different task arrival intensities. In contrast, as shown in Fig. 2(a), the binary reward feedback of UCB1 and the exploration-exploitation strategy of BANDIT-EPSILON cannot find a good offloading policy in the dynamic network. The lack of exploration of GREEDY makes it fall into local optimal, with the worst results eventually.

Another task-generating pattern is adjusting the number of UAVs in the MEC system. The number of UAVs affects the load capacity of the edge server in the long run. Fig. 3(a) and Fig. 3(b) demonstrate the cumulative system cost and the satisfaction rate under different numbers of UAVs, respectively. Here, we set the task intensity of UAVs to 30 % at an arbitrary time. Furthermore, compared to the simulation in Fig. 2(a), we evaluate the service deadline from 10 to 15 seconds. In fact, we want to evaluate the performance under different continuous loads from UAVs with stringent deadlines. As the number of UAVs increases, BANDIT-SCH can still outperform other baselines and is close to the performance of UPPER. The key point is that the custom reward and the one-side confidence let BANDIT-SCH handle the higher continuous loading in the system and satisfy requests of UAVs with stringent deadlines.

C. The Convergence of Learning Offloading Policy

To analyze the convergence degree of the MAB-related schemes, we combine the two figures of Fig. 2(a) and Fig. 3(a) jointly. Finding the policy in the short dispatch rounds is difficult for the bandit-related schemes since the agent is still in its initial exploration phase. Unlike other MAB approaches, taking (16) as the learning direction, the binary reward of UCB1 cannot adapt to insufficient requests and fails in a highly dynamic MEC system. However, the BANDIT-SCH scheme can achieve the fast-adaptation and provide feasible solutions in these two stochastic offloading patterns. With task intensity increasing, although more essential data is inevitably sacrificed in BANDIT-SCH compared with UPPER, the cumulative system cost will increase marginally. We're excited to observe that BANDIT-SCH maintains the same satisfaction rate as UPPER in Fig. 2.

V. Conclusion

This paper investigates the MEC-assisted task offloading problem under various deadline constraints with multi-UAVs to minimize total MEC system data loss costs. To provide various PA tasks in uncertain real-world environments, we proposed an RL-based task offloading mechanism, BANDIT-SCH, which includes a deadline-aware dispatch policy DABD and a minimized scheduling MQ-SCH. The *custom deadline-aware reward* and the *one-sided confidence interval* let BANDIT-SCH perform well in exploration and exploitation to adapt to stochastic offloading and dynamic system loading. The experiment results show that the performance of the BANDIT-SCH is approximate to the UPPER and can preserve acceptable MEC system cost and satisfaction rate in precision agriculture. In future work, we will add UAV-mounted edges to assist various task offloading and explore how to design an RL-based strategy in such advanced MEC-assisted edge systems for realizing PA services.

References

[1] P. K. Reddy Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, and Q.-V. Pham, "Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 608–17 619, Aug 2021.

[2] N. Ahmed, D. De, and I. Hussain, "Internet of things (IoT) for smart precision agriculture and farming in rural areas," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, Dec 2018.

[3] J. Engebretson. Private wireless network comes to the farm, enabling precision agriculture. https://www.telecompetitor.com/private-wireless-network-comes-to-the-farm-enabling-precision-agriculture/. [Online; accessed: 2022-2-14].

[4] S. Cheng, Z. Xu, X. Li, X. Wu, Q. Fan, X. Wang, and V. C. Leung, "Task offloading for automatic speech recognition in edge-cloud computing based mobile networks," in *IEEE ISCC*, Rennes, France, 2020, pp. 1–6.

[5] B. Dab, N. Aitsaadi, and R. Langar, "Joint optimization of offloading and resource allocation scheme for mobile edge computing," in *IEEE WCNC*, Marrakesh, Morocco, 2019, pp. 1–7.

[6] T. Zhang, Y.-H. Chiang, C. Borcea, and Y. Ji, "Learning-based offloading of tasks with diverse delay sensitivities for mobile edge computing," in *IEEE GLOBECOM*, Waikoloa, HI, USA, 2019, pp. 1–6.

[7] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *IEEE INFOCOM*, Paris, France, 2019, pp. 2287–2295.

[8] S. Lee, S. Lee, and S.-S. Lee, "Deadline-aware task scheduling for IoT applications in collaborative edge computing," *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2175–2179, Oct 2021.

[9] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 196–210, Jan 2022.

[10] R. Duan, J. Wang, J. Du, C. Jiang, T. Bai, and Y. Ren, "Power-delay trade-off for heterogenous cloud enabled multi-uav systems," in *IEEE ICC*, Shanghai, China, 2019, pp. 1–6.

[11] A. Gao, Q. Wang, Y. Hu, and W. Duan, "An offloading optimization scheme for multi-uav aided network in mobile computing," in *IWCMC*, Limassol, Cyprus, 2020, pp. 1468–1473.

[12] L. Wang, P. Huang, K. Wang, G. Zhang, L. Zhang, N. Aslam, and K. Yang, "RL-based user association and resource allocation for multi-uav enabled mec," in *IWCMC*, Tangier, Morocco, 2019, pp. 741–746.

[13] P. Yan and S. Choudhury, "Optimizing mobile edge computing multi-level task offloading via deep reinforcement learning," in *IEEE ICC*, Dublin, Ireland, 2020, pp. 1–7.

[14] C.-L. Wu, T.-C. Chiu, C.-Y. Wang, and A.-C. Pang, "Mobility-aware deep reinforcement learning with glimpse mobility prediction in edge computing," in *IEEE ICC*, Dublin, Ireland, 2020, pp. 1–7.

[15] H. Liao, Z. Zhou, S. Mumtaz, and J. Rodriguez, "Robust task offloading for iot fog computing under information asymmetry and information uncertainty," in *IEEE ICC*, Shanghai, China, 2019, pp. 1–6.

[16] B. Wu, T. Chen, K. Yang, and X. Wang, "Edge-centric bandit learning for task-offloading allocations in multi-rat heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3702–3714, Apr 2021.

[17] S. Sun, T. S. Rappaport, S. Rangan, T. A. Thomas, A. Ghosh, I. Z. Kovacs, I. Rodriguez, O. Koymen, A. Partyka, and J. Jarvelainen, "Propagation path loss models for 5g urban micro- and macro-cellular scenarios," in *IEEE VTC Spring*, Nanjing, China, 2016, pp. 1–6.

[18] Y. Ma, W. Liang, M. Huang, W. Xu, and S. Guo, "Virtual network function service provisioning in MEC via trading off the usages between computing and communication resources," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2949–2963, Dec 2022.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002.