Schedule or Wait: Age-Minimization for IoT Big Data Processing in MEC via Online Learning

Zichuan Xu[†], Wenhao Ren[†], Weifa Liang[‡], Wenzheng Xu[§*], Qiufen Xia[$], Pan Zhou[%], Mingchu Li[†].
[†] School of Software, Dalian University of Technology, Dalian, China, 116621.
[‡] Department of Computer Science, City University of Hong Kong, Hong Kong.
[§] College of Computer Science, Sichuan University, Chengdu, Sichuan, China, 610065.
[$] International School of Information Science and Engineering, Dalian University of Technology, Dalian, China, 116621.
[%] Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering,
Huazhong University of Science and Technology, Wuhan, China, 430074.
Emails: z.xu@dlut.edu.cn, hangvane@mail.dlut.edu.cn, weifa.liang@cityu.edu.hk, wenzheng.xu@scu.edu.cn, qiufenxia@dlut.edu.cn,
panzhou@hust.edu.cn, mingchul@dlut.edu.cn.
[*] Corresponding author: Wenzheng Xu.

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796718

***Abstract*—The age of data (AoD) is identified as one of the most novel and important metrics to measure the quality of big data analytics for Internet-of-Things (IoT) applications. Meanwhile, mobile edge computing (MEC) is envisioned as an enabling technology to minimize the AoD of IoT applications by processing the data in edge servers close to IoT devices. In this paper, we study the AoD minimization problem for IoT big data processing in MEC networks. We first propose an exact solution for the problem by formulating it as an Integer Linear Program (ILP). We then propose an efficient heuristic for the offline AoD minimization problem. We also devise an approximation algorithm with a provable approximation ratio for a special case of the problem, by leveraging the parametric rounding technique. We thirdly develop an online learning algorithm with a bounded regret for the online AoD minimization problem under dynamic arrivals of IoT requests and uncertain network delay assumptions, by adopting the Multi-Armed Bandit (MAB) technique. We finally evaluate the performance of the proposed algorithms by extensive simulations and implementations in a real test-bed. Results show that the proposed algorithms outperform existing approaches by reducing the AoD around 10%.**

***Index Terms*—Mobile edge cloud; Big data processing; Age of data; Approximation algorithm; Online learning.**

I. Introduction

With the fast development of 5G, various Internet of Things (IoT) devices including sensors, sensory devices and wearables, are connecting with each other. Due to the nature of such machine-to-machine communications, gigabytes and terabytes of information will whizz between IoT devices at an unprecedented speed. This requires big data technologies to analyze the collected, *IoT big data*, then obtain valuable insights for decision-making. One distinctive feature of IoT big data is that the quality of analytics depends on the age of data (AoD), where the AoD of a piece of data is the elapsed time since the data generation. This is because the newly generated data is typically "hot" and frequently accessed. As time goes on, the data becomes cold and is infrequently or never accessed. Therefore, big data collected from IoT devices has to be analyzed in a timely manner such that the AoD is minimized. For example, Google uses AoD to indicate the data retention time in the dataflow pipeline and the latency of data statistics [10], [11].

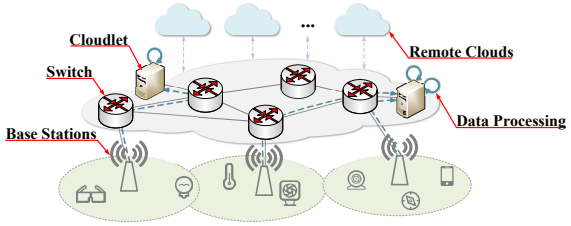


Fig. 1. An example of the hierarchical MEC network.

Conventional big data analytics deals with big data processing in remote clouds [22], [50]. Since remote clouds are far from IoT devices, they result in significant communication delays of data processing. The data will become stale and not be valid anymore for a real-time response. Mobile Edge Computing (MEC) that deploys cloudlets in the close proximity of IoT devices can guarantee the timeliness of data processing [25]. To meet both the timelineness and huge resource demand of big data processing, we consider a hierarchical MEC network that consists of both local cloudlets and remote clouds, which is operated by an IoT service provider, as shown in Fig. 1. In the MEC network, multiple IoT devices measure the physical environment of a region, and all measurements are required to send to either cloudlets or remote clouds of the MEC network through their nearby base stations. We investigate the age-aware big data processing for IoT applications in such a hierarchical MEC network to minimize the AoD of IoT big data analytics.

Minimizing the AoD of IoT big data analytics in a hierarchical MEC network poses several fundamental challenges.

First, the AoD minimization involves complicated interplays among time elapsed before transmission, data transmission delay, and processing delay. On the generation of a dataset, we need to decide schedule the processing of the dataset or wait. Specifically, if a dataset is scheduled for transmission immediately after its generation, the cloudlets or backhaul links may be too congested to process the dataset. Instead, if a dataset is postponed for scheduling later, we may increase its AoD. However, the dataset may be processed at a very fast pace, offsetting the increased AoD due to waiting for scheduling and transmission delays. How to jointly consider the generation time of datasets, resource availabilities of cloudlets, and transmission

latencies in IoT big data processing is challenging.

Second, IoT service requests usually arrive into the system dynamically, and the network delays are not known in advance. How to make use of deep learning techniques to assist scheduling of big data analytics requests with uncertain network delays while minimizing the regret of decision-making?

Third, each IoT service provider has its own running budget on energy, transmission, and processing costs, to avoid unpredictable cost surges in unexpected situations. Furthermore, cloudlets and remote clouds in the MEC network have computing resource capacity constraints. It is challenging to jointly consider the budget and resource capacity constraints while minimizing the AoD of IoT big data analytics.

Although there are extensive studies of big data processing in remote clouds or MEC networks [2], [7], [8], [12], [21], [22], [26], [27], [31], [39], [41], [42], [43], [46], [48], most of them ignored the AoD requirements of IoT big data processing. Closely related studies to our work in this paper are the ones on Age-of-Information (AoI) minimization in networks [5], [9], [15], [18], [20], [32], [33], [37], [40], [49]. Most of these studies however do not consider the uncertain delays of the MEC network. Also, they aim to minimize the age of packet-level data instead of big data.

To the best of our knowledge, we are the first to consider the age-aware IoT big data processing in an MEC network with uncertain network delays. We formulate a novel optimization problem, and develop approximate solutions with performance guarantees to it. To handle the uncertainties in the MEC network, we devise a novel online learning algorithm with a bounded regret. The main contributions of this paper include:

- We define the AoD minimization problem for IoT applications in a hierarchical MEC network and formulate an Integer Linear Program (ILP) solution to the problem.
- We develop an efficient heuristic for the offline AoD minimization problem. We also propose an approximation algorithm with an approximation ratio for a special case of the problem, by leveraging parametric LP relaxation [1].
- We devise an online learning algorithm for the online AoD minimization problem without the knowledge of future request arrivals and network delays, by leveraging the multi-armed bandit (MAB) technique. We also analyze the regret bound of the online learning algorithm.
- We evaluate the performance of the proposed algorithms by extensive simulations and implementations in a test-bed. Results show that the proposed algorithms outperform existing studies by reducing the AoD around 10%.

The rest of this paper is organized as follows. Section II summarizes related studies. Section III introduces the system model and defines the AoD minimization problem. Section IV gives an ILP solution, and Section V proposes a heuristic and an approximation algorithm for the offline AoD minimization. Section VI proposes an online learning algorithm for the online AoD minimization problem. Section VII evaluates the performance of the proposed algorithms by simulations and real implementations, and Section VIII concludes the paper.

II. Related work

Big data processing has attracted much attention in literature. Most existing studies focused on big data placement and query evaluation optimization for big data analytics in remote clouds [13], [17], [19], [28], [38], [50]. These proposed methods however may not be applicable for IoT applications that require real-time analysis on collected datasets due to long data transmission delays between IoT devices and clouds. There are several studies on big data processing in MEC networks [2], [8], [22], [26], [27], [39], [41], [43], [46]. For example, Aujla *et al.* [2] investigated the problem of energy-aware network traffic flow scheduling for big data analytics in an MEC network. Cheng *et al.* [4] investigated the problem of task scheduling for large-scale big data processing, by proposing a multi-agent deep reinforcement learning model. Lu *et al.* [22] proposed a method to predict MapReduce performance in multiple edge clouds for IoT big data processing. Zhou *et al.* [50] proposed a neural network based on Map-Reduce on cloud computing cluster for big data processing of multimedia communication. However, most of these studies did not incorporate the AoD requirements of IoT applications into consideration.

Existing studies on AoI minimization normally considered a single data source or the processing in a single edge server [5], [7], [12], [15], [18], [21], [31], [35], [44], [48]. For example, Corneo *et al.* [7] proposed grouping algorithms to strive for a balance between AoI and the number of sensor update transmissions. Song *et al.* [31] employed age of task to evaluate the temporal value of computation tasks, and proposed a light-weight, age-based task scheduling and computation offloading algorithm. In wireless ad hoc networks, Lu *et al.* [21] adopted a virtual-queue technique to design a scheduling algorithm that jointly ensures the timely throughput and AoI. Zhong *et al.* [48] proposed a greedy traffic scheduling policy to minimize the AoI and proved the optimality of the algorithm.

There are also studies on age minimization of packet-level data instead of big data [3], [9], [29], [32], [33], [34], [37], [47], [49]. Sun *et al.* [32] modeled the sampling process as a queue and formulated the problem as a constrained Markov decision process. Zhong *et al.* [49] studied the age-aware multicasting problem where data is sent to multiple nodes by proposing a priority-queue-based optimization algorithm. Zhuang *et al.* [47] investigated the routing in a communication network to minimize AoI, by considering time-varying propagation delays. Bedewy *et al.* [3] investigated scheduling policies that minimize the AoI in single-hop queuing systems by applying the last-generated first-serve scheduling policy. These mentioned methods considered the AoI at the packet level and cannot be applied to the IoT data processing in MEC networks.

III. Preliminaries

A. System Model and AoD

We consider a hierarchical MEC network $G = (\mathcal{BS} \cup \mathcal{CL} \cup \mathcal{DC}, E)$ operated by an IoT service provider, which consists of a set of cloudlets $\mathcal{CL}$ and a set of remote datacenters $\mathcal{DC}$, as shown in Fig 1. Denote by CL_i and DC_j a cloudlet

and datacenter, respectively. Each cloudlet $CL_i \in \mathcal{CL}$ has a few commodity servers to implement various user requests. Denote by $C(CL_i)$ and $C(DC_j)$, the computing capacities to process the data of IoT devices on cloudlet CL_i and datacenter DC_j, respectively. IoT devices access the cloudlets via base stations in $\mathcal{BS}$, where an IoT device may be in the ranges of multiple base stations. Let bs_k be a base station in $\mathcal{BS}$. E is a set of links interconnecting cloudlets and datacenters.

IoT devices continuously generate data and IoT applications issue requests to process the generated data. Denote by dv_m, an IoT device indexed by m. Each dv_m continuously generates data for a given monitoring time period, so this time period is split into equal time slots. Denote by $r_m(t)$ the request of IoT device dv_m at time slot t, and let $R(t)$ be the set of requests by all IoT devices at time slot t. Denote by $ds_{m,t}$ the dataset generated by an IoT device dv_m at time slot t. A request $r_m(t)$ is issued as soon as the generation of a dataset $ds_{m,t}$ at time slot t. Suppose the current time slot is τ, the AoD of $r_m(t)$ is defined as $\tau - t$. For simplicity, we consider a cloudlet or a datacenter as a *potential location* to implement a request. Let $\mathcal{LOC}$ be the set of potential locations, i.e., $\mathcal{LOC} = \mathcal{CL} \cup \mathcal{DC}$. Let LOC_l be a potential location in $\mathcal{LOC}$, and the computing capacity of LOC_l is denoted by $C(LOC_l)$.

B. Cost Model

The IoT service provider of an MEC network has its own IoT devices and leases resources from cloud service providers to process its IoT big data. We thus consider the energy, transmission, processing, and storage costs as monetary costs.

The transmission of the dataset $ds_{m,t}$ of request $r_m(t)$ from its IoT device dv_m to a base station consumes energy of dv_m. Assuming that the bandwidth of the IoT device dv_m is bw_m, the achieved data rate $W_{m,k}$ via the wireless channel between the IoT device and base station bs_k is calculated by

$$W_{m,k} = bw_m \log_2 \big(1 + (p_m \cdot h_{m,k})/(\sigma^2 + I_m)\big), \quad (1)$$

where σ^2 is the noise power of IoT devices and I_m is the inter-cell interference power, $h_{m,k}$ is the channel gain between IoT device dv_m and base station bs_k, and p_m is the transmission power of IoT device dv_m [14].

The energy cost spent in data transmission from dv_m to base station bs_k thus can be calculated by

$$c^{energy}_{m,k,t} = w_e \cdot p_m \cdot (|ds_{m,t}|/W_{m,k}), \quad (2)$$

where w_e is a given constant of the monetary cost of per unit energy consumption and $|ds_{m,t}|$ is the data volume of $ds_{m,t}$.

The dataset $ds_{m,t}$ of dv_m will be placed at a cloudlet or a remote datacenter via a base station bs_k. Let $c_{k,l}$ be the cost of transmitting a unit amount of data via the link/path between base station bs_k and LOC_l. The cost for transferring dataset $ds_m(t)$ from bs_k of IoT device dv_m to LOC_l is

$$c^{trans}_{m,k,l,t} = c_{k,l} \cdot |ds_{m,t}|. \quad (3)$$

Following existing studies [24], we assume that processing a unit of dataset at location LOC_l consumes a unit amount δ_l of computing resource, where δ_l is a given constant. The total amount of computing resource assigned for processing dataset $ds_{m,t}$ on LOC_l thus is $\delta_l \cdot |ds_{m,t}|$. Denote by c_l the cost of using a unit amount of computing resource in LOC_l. The cost of processing dataset $ds_{m,t}$ in LOC_l thus is

$$c^{proc}_{m,l,t} = c_l \cdot \delta_l \cdot |ds_{m,t}|. \quad (4)$$

Storing a unit of data in location LOC_l incurs a cost of ϕ_l, where ϕ_l is a constant. Let $c^{sto}_{m,l,t}$ be the cost of storing dataset $ds_{m,t}$ in LOC_l, then

$$c^{sto}_{m,l,t} = \phi_l \cdot |ds_{m,t}|. \quad (5)$$

C. Delay Model

Suppose that $d_{m,k}$ is the delay of transferring a unit of data along the wireless channel of base station bs_k, and $d_{k,l}$ is the delay of transferring a unit of data from bs_k to location LOC_l. Let d_l be the delay of processing a unit amount of data in location LOC_l. Let $d_{m,k} = |ds_{m,t}|/W_{m,k}$, then the delay $d_{m,k,l,t}$ of processing dataset $ds_{m,t}$ in LOC_l via bs_k is

$$d_{m,k,l,t} = (d_{m,k} + d_{k,l} + d_l) \cdot |ds_{m,t}|. \quad (6)$$

D. Problem Definition

Given a hierarchical MEC network G and a finite time horizon T that is divided into T equal time slots, a set of IoT devices, a set of requests $R(t) = \{r_m(t) \,|\, 1 \le m \le M, 1 \le t \le T\}$ at time slot t, assuming that there is a given budget B_m for each IoT device dv_m to store, transmit, and process its datasets, the *AoD minimization problem in a hierarchical MEC network* is to minimize the total age of all processed datasets by the MEC network, subject to the computing capacity on each potential location $LOC_l \in \{\mathcal{CL} \cup \mathcal{DC}\}$ and the budget B_m on each IoT device dv_m on the costs of storing, transmitting, and processing its datasets within time horizon T.

IV. ILP Formulation

We now propose an exact solution to the AoD minimization problem in a hierarchical MEC network, by formulating the problem to an Integer Linear Program (ILP). Recall that the dataset $ds_{m,t}$ is generated at time slot t by IoT device dv_m. Let τ be the time that $ds_{m,t}$ is processed by either a cloudlet or a datacenter. Dataset $ds_{m,t}$ may not be scheduled for processing immediately after its generation. The waiting time from its generation to the time slot of its processing is $\tau - t$. Once scheduled, dataset $ds_{m,t}$ will be transferred to a cloudlet or a datacenter. The transmission and processing time of $ds_{m,t}$ thus contributes to the AoD of $ds_{m,t}$.

Denote by $x_{\tau,l,m,k}(t)$ the binary decision variable that indicates whether LOC_l is used to process $ds_{m,t}$ via base station bs_k at time slot τ. The AoD of $ds_{m,t}$ thus is

$$DA_{m,t} = \sum_{\tau=t}^{T} \sum_{bs_k \in \mathcal{BS}} \sum_{LOC_l \in \mathcal{LOC}} x_{\tau,l,m,k}(t) \cdot (d_{m,k,l,t} + \tau - t). \quad (7)$$

The objective of the problem is to

$$\textbf{ILP}: \min \quad \sum\nolimits_{m=1}^{|R|} \sum\nolimits_{t=1}^{T} DA_{m,t}, \quad (8)$$

subject to the following constraints,

$$\sum_{bs_k \in \mathcal{BS}} \sum_{LOC_l \in \mathcal{LOC}} x_{\tau,l,m,k}(t) = 0, \quad \forall r_m(t) \in R, \ \forall \tau < t. \tag{9}$$

$$\sum_{bs_k \in \mathcal{BS}} \sum_{LOC_l \in \mathcal{LOC}} x_{\tau,l,m,k}(t) = 1, \forall r_m(t) \in R, \ \forall \tau \in [1, T]. \tag{10}$$

$$\sum\nolimits_{r_m(t) \in R} \sum\nolimits_{bs_k \in \mathcal{BS}} x_{\tau,l,m,k}(t) \cdot \delta_l \cdot |ds_{m,t}| \leq C(LOC_l), \quad \forall LOC_l \in \mathcal{LOC}, \ \forall 1 \leq \tau \leq T. \tag{11}$$

$$\sum_{bs_k \in \mathcal{BS}} \sum_{LOC_l \in \mathcal{LOC}} x_{\tau,l,m,k}(t) \cdot c_{m,k,l,t} \leq B_m, \forall r_m(t). \tag{12}$$

$$x_{\tau,l,m,k}(t) \in \{0, 1\}, \tag{13}$$

where $c_{m,k,l,t} = c^{energy}_{m,k,t} + c^{trans}_{m,k,l,t} + c^{sto}_{m,l,t} + c^{proc}_{m,l,t}$.

Constraint (9) ensures that dataset $ds_{m,t}$ of request $r_m(t)$ cannot be scheduled before its generation. Constraint (10) guarantees that dataset $ds_{m,t}$ is placed to either a cloudlet or a remote datacenter via a base station. Constraint (11) says that the computing capacity of a location is no less than the total resource demand of requests assigned to it. Constraint (12) ensures that the budget of each IoT device is not violated.

V. Heuristic and Approximation Algorithms for the Offline AoD Minimization Problem

The ILP solution may not be scalable when the problem size is large. In this section we instead devise an efficient heuristic for the offline age of data minimization problem. We also propose an approximation algorithm with an approximation ratio for the problem without the budget constraint.

A. Heuristic Algorithm

Our basic idea is to relax the **ILP** to a Linear Program (**LP**). Specifically, Constraint (13) is relaxed to

$$0 \leq x_{\tau,l,m,k}(t) \leq 1. \tag{14}$$

We then obtain a fractional solution to the **LP** in polynomial time. Denote by x^* the optimal fractional solution to the **LP**. The fractional solution represents the scheduling of dataset $ds_m(t)$ into different time slots and splitting it to multiple locations (either cloudlets or datacenters). This solution however is infeasible to the original problem because each dataset can only be scheduled in a single time slot and assigned to a single location for processing. We thus need to modify the fractional solution to a feasible, integral solution to the original problem. To this end, we first obtain a set of candidate time slots and candidate locations to process each dataset $ds_{m,t}$ based on the fractional solution. Specifically, we set a threshold η on the fractional values of each variable. The time slots, cloudlets or datacenters with their corresponding fractional values greater than η are considered as the candidate time slots, candidate cloudlets or candidate datacenters, respectively. Denote by $\mathcal{T}_{m,t}$ the set of candidate time slots for dataset $ds_{m,t}$, i.e.,

$$\mathcal{T}_{m,t} = \{\tau \mid x_{\tau,l,m,k}(t) > \eta\}. \tag{15}$$

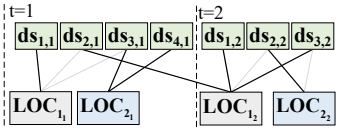


Fig. 2. An example of algorithm Heu.

Algorithm 1 Heu

Input: $G = (\mathcal{BS} \cup \mathcal{LOC}, E)$, a set of requests.
Output: Request assignments.

```
Solve LP and obtain a fractional solution x*;
Initialize V_a = {ds_{m,t} | ∀m, ∀t ≤ T};
Sort V_a into a non-decreasing order in terms of the data volumes of datasets;
while V_a ≠ {} do
    Initialize T_{m,t}, LOC_{m,t} by Eq.(15), (16);
    Initialize E' = {} and V_b = {};
    Add T virtual locations for each location LOC_l ∈ LOC_{m,t} to V_b;
    for each ds_{m,t} ∈ V_a and each LOC_{l,τ} ∈ V_b do
        Add an edge ⟨ds_{m,t}, LOC_{l,τ}⟩ with weight DA_{m,t} to E' if
        LOC_l ∈ LOC_{m,t} and τ ∈ T_{m,t} and processing ds_{m,t} does
        not violate constraints (9) – (12);
    Find a minimum weight maximum matching M' in G' = {V_a, V_b; E'};
    Update schedule by M'; V_a = {ds_{m,t} | ∀ds_{m,t} is not scheduled};
```

Similarly, the candidate location set for each dataset $ds_{m,t}$ is

$$\mathcal{LOC}_{m,t} = \{LOC_l \mid x_{\tau,l,m,k}(t) > \eta\}. \tag{16}$$

We then round the fractional solution to an integral one by transferring the problem to a minimum weight maximum matching problem in a bipartite graph. For each location LOC_l let G' be the constructed bipartite graph. As shown in Fig. 2, we have two sets of nodes, i.e., V_a and V_b. We add each dataset $ds_{m,t}$ to set V_a. We then create T virtual locations for each location, where each time slot τ has a virtual location. Denote by $LOC_{l,\tau}$ the τth virtual location at time slot τ. Those virtual nodes are added to V_b.

We sort all datasets into non-decreasing order of their volume, and consider the sorted datasets in turn. For each dataset $ds_{m,t}$ in the sorted list, we add an edge between $ds_{m,t}$ and $LOC_{l,\tau}$ if and only if (1) τ is in set $\mathcal{T}_{m,t}$ and LOC_l is within set $\mathcal{LOC}_{m,t}$; (2) processing $ds_{m,t}$ does not violate the residual computing resource capacity of LOC_l; and (3) the residual budget B_m of IoT device dv_m is not violated. The weight of the edge is set to the AoD of $ds_{m,t}$ if it is processed at time slot τ within location LOC_l, i.e., $w(\langle ds_{m,t}, LOC_{l,\tau}\rangle) = DA_{m,t}$. We then find a minimum weight maximum matching in the bipartite graph G', which corresponds an assignment of a subset of datasets to cloudlets and datacenters. This procedure continues until all datasets are scheduled.

The heuristic is detailed in **Algorithm** 1, referred to as algorithm Heu.

B. Approximation Algorithm for a Special Case of the Problem

We now consider a special case of the problem when $\tau = t$, where all datasets generated at this time slot are required to be scheduled immediately. We thus replace $x_{\tau,l,m,k}(t)$ in the **ILP** with $x_{l,m,k}(t)$, and obtain **ILP-SC** as follows.

$$\mathbf{ILP-SC}: \ \min \quad \sum\nolimits_{m=1}^{|R|} \sum\nolimits_{l,k} x_{l,m,k}(t) \cdot d_{m,k,l,t}, \tag{17}$$

subject to Constraints (9) – (12) and $x_{l,m,k}(t) \in \{0,1\}$.

The basic idea of the proposed approximation algorithm is similar to that of algorithm Heu, which adopts an LP rounding technique. That is, we first come up with an appropriate relaxation. A natural relaxation of the **ILP** is shown in Eq. (14). This, however, may reduce the quality of the solution. The reason is that the fractional solution due to Eq. (14) may lead a dataset being "split" into many small pieces. Such cloudlets or datacenters with small fractional values of their corresponding decision variables may have higher processing costs. To avoid this, we adopt the parametric LP relaxation method [1] by relaxing the special case of the problem to an LP that disallows the assignment of datasets to cloudlets or datacenters with their resource consumptions no more than a threshold res^{th}.

Let **LP-SC**(res^{th}) be the parametric LP relaxation. Its objective is the same as **ILP-SC** shown in Eq. (17), subject to Constraints (9) – (12), and the following constraints,

$$x_{l,m,k}(t)=0, \text{ for each } ds_{m,t} \text{ with } |ds_{m,t}|\cdot\delta_l > res^{th}, \quad (18)$$

$$0 \le x_{l,m,k}(t) \le 1. \quad (19)$$

We then obtain a fractional solution x'^* by solving the **LP-SC**(res^{th}). Based on x'^*, we construct another bipartite graph $G'' = (V_a'', V_b''; E'')$, where V_a'' and V_b'' are two sets and E'' is the set of links that interconnect these two sets.

Denote by $deg_l = \sum_{m=1}^{|R|} x'_{l,m,k}(t)$ the fractional assignment of location LOC_l, i.e., the fractional numbers of datasets that are assigned to location LOC_l. Let $\kappa_l = \lceil deg_l \rceil$. In G'', we create κ_l virtual locations for LOC_l, i.e., κ_l copies of LOC_l, which are added to V_b''. This guarantees that LOC_l is assigned with κ_l or $(\kappa_l - 1)$ datasets for processing. If we split the assignment of each dataset evenly among the κ_l copies of location LOC_l, we can obtain a fractional solution to the minimum weight maximum matching problem, as shown in algorithm Heu. We can also obtain an integral solution with an AoD that is almost the same as that of the fractional solution. But we cannot guarantee how much the capacity constraint of location LOC_l and the budget of IoT device dv_m are violated.

To bound the violations on the computing resource capacities and the budget constraints, we assign datasets with roughly similar data volumes to location LOC_l. To this end, we sort the datasets into non-increasing order of their data volume and go over each dataset in turn. Specifically, we start with the first copy of LOC_l, i.e., LOC_{l_1}. We keep adding an edge between $ds_{m,t}$ and LOC_{l_1} as long as $\sum_{m=1}^{|R|} x'_{l_1,m,k}(t)$ is smaller than 1. The edge weight is set as the AoD with an assignment of fractional volume of $ds_{m,t}$, i.e., $x'_{l_1,m,k}(t) \cdot d_{m,k,l,t}$. For simplicity, such an assignment of fractional volume of a dataset is referred to as *fractional assignment*. When we encounter a dataset that would push $\sum_{m=1}^{|R|} x'_{l_1,m,k}(t)$ over 1, we add an edge to E'' with a fractional assignment such that $\sum_{m=1}^{|R|} x'_{l_1,m,k}(t) = 1$. We also add an edge $\langle ds_{m,t}, LOC_{l_2} \rangle$ to E'' with fractional assignment $x'_{l_2,m,k}(t) = x'_{l,m,k}(t) - x'_{l_1,m,k}(t)$. This procedure continues until all datasets are considered. Having constructed the bipartite graph G'', we find a minimum weight maximum matching M'' in G''. The detailed algorithm is shown in Algorithm 2, which is referred to as algorithm Appro.

Algorithm 2 Appro

Input: $G = (\mathcal{BS} \cup \mathcal{LOC}, E)$, a set of requests generated in time slot t.
Output: Request assignments in time slot τ.

Solve **LP-SC**(res^{th}) and obtain a fractional solution x'^*;
Add κ_l $(= \lceil deg_l \rceil)$ virtual nodes to V_b'' for each LOC_l;
Add all datasets into V_a'' and sort V_a'' into a non-increasing order in terms of the data volumes of datasets;
for each $LOC_l \in \mathcal{LOC}$ **do**
  $\kappa = 1$;
  for each $ds_{m,t} \in V_a$ **do**
    if $\sum_{m=1}^{|R|} x'_{l_\kappa,m,k}(t) + x'_{l,m,k}(t) > 1$ **then**
      Add an edge $\langle ds_{m,t}, LOC_{l_\kappa} \rangle$ with weight $x'_{l_\kappa,m,k}(t) = 1 - \sum_{m=1}^{|R|} x'_{l_\kappa,m,k}(t)$ to E'';
      Add an edge $\langle ds_{m,t}, LOC_{l_{\kappa+1}} \rangle$ with weight $x'_{l_{\kappa+1},m,k}(t) = x'_{l,m,k}(t) - x'_{l_\kappa,m,k}(t)$ to E'';
      $\kappa = \kappa + 1$;
    else
      Add an edge $\langle ds_{m,t}, LOC_{l_\kappa} \rangle$ with weight $x'_{l_\kappa,m,k}(t) = x'_{l,m,k}(t)$ to E'';
Find a minimum weight maximum matching M'' in $G' = \{V_a'', V_b''; E''\}$;
Update schedule by M'';

C. Algorithm Analysis

Theorem 1: Algorithm Heu delivers a feasible solution to the offline AoD minimization problem in an MEC network G in $O(\sum_{t=1}^{T} |R(t)| \cdot (\sum_{t=1}^{T} |R(t)|(1 + |\mathcal{BS}| \cdot |\mathcal{LOC}|))^{\omega} L)$ time, where ω is the exponent of matrix multiplication and the current value of $\omega \approx 2.37$, and L is the input bits [6].

Proof: We first show the solution feasibility of algorithm Heu. In the construction of bipartite graph G'', the only edge is between dataset $ds_{m,t}$ and a candidate virtual location $LOC_{l,t}$, when $\tau \ge t$ and location LOC_l can meet both the computing resource demand of $ds_{m,t}$ and the budget B_m of dv_m. The solution thus is feasible.

The running time of algorithm Heu is mainly due to the solving of **LP** and finding the maximum matching. The analysis is straightforward; omitted due to the space limitation. ■

Lemma 1: Algorithm Appro violates the capacity of each cloudlet or datacenter by a ratio of $(1+\epsilon)$, where $\epsilon = \frac{res^{th}}{C(LOC_l)}$ is a small given constant with $\frac{ds_{min}\delta_{min}}{C(LOC_l)} < \epsilon < \frac{ds_{max}\delta_{max}}{C(LOC_l)}$ and $ds_{max} = \max_m\{ds_{m,t}\}$, $\delta_{max} = \max_l\{\delta_l\}$, respectively.

Proof: For each location LOC_l, we split the contribution to its capacity violation to two components: the first virtual location LOC_{l_1} and all the rest virtual locations. From the construction of bipartite graph G'' in algorithm Appro, we know that LOC_{l_1} has an accumulative assignment of 1 in the fractional solution, meaning that it can be assigned to at most one dataset in the resulting integral matching. Therefore, the contributed computing resource consumption is at most res^{th}.

For each virtual location LOC_{l_o}, we use res_{l_o} and $res_{w(l_o)}$ to denote the minimum and maximum resource consumptions of any datasets that can be assigned to LOC_{l_o}, where $2 \le l_o, w(l_o) \le \mathcal{O}_i$ with $\mathcal{O}_i$ denoting the number of virtual locations of LOC_l in bipartite graph G''. In the worst case, we assign the datasets with the maximum volumes to all virtual locations LOC_{l_o}. The contributed computing resource consumption of the datasets that are assigned to virtual locations in $\{LOC_{l_o} \mid 2 \le l_o \le \mathcal{O}_l\}$ is at most $\sum_{l_o=2}^{\mathcal{O}_l} res_{w(l_o)}$. In

algorithm Appro, we go over the datasets one by one in non-increasing order of their data volume. Thus, the minimum dataset that is assigned to the previous virtual location (i.e., $LOC_{l_{o-1}}$) is higher than the maximum dataset assigned to the l_oth virtual location. We thus have $res_{w(l_o)} \leq res_{l_{o-1}}$ and

$$\sum_{l_o=2}^{\mathcal{O}_l} res_{w(l_o)} \leq \sum_{l_o=1}^{\mathcal{O}_l-1} res_{l_o}, \tag{20}$$

$$\leq \sum_{l_o=1}^{\mathcal{O}_l-1} \sum_{m=1}^{|R|} x_{l_o,m,k}(t) res_{l_o} \leq C(LOC_l). \tag{21}$$

This concludes that the capacity of LOC_l is violated by at most $\frac{C(LOC_l)+res^{th}}{C(LOC_l)}$. Assuming that the maximum resource consumption of any dataset is far less than the capacity of a cloudlet, we have $(C(LOC_l)+res^{th})/(C(LOC_l)) \leq (1+\epsilon)C(LOC_l)$, where $\epsilon = res^{th}/C(LOC_l)$ is a given constant with $(ds_{min}\delta_{min})/(C(LOC_l)) < \epsilon < (ds_{max}\delta_{max})/(C(LOC_l))$. ■

Lemma 2: Let $ds_{max} = \max_{m=1}^{|R|}\{|ds_{m,t}|\}$, $ds_{min} = \min_{m=1}^{|R|}\{|ds_{m,t}|\}$, $\delta_{max} = \max_l\{\delta_l\}$ and $\delta_{min} = \min_l\{\delta_l\}$, and assume that $ds_{min} \cdot \delta_{min} \leq res^{th} \leq ds_{max} \cdot \delta_{max}$. Algorithm Appro violates the budget B_m of each IoT device dv_m by ζ/δ_{min} times, where $\zeta = ds_{max}/ds_{min}$.

Proof: As shown in Lemma 1, for each location LOC_l, the maximum resource consumption that can be assigned to virtual location LOC_{l_1} is res^{th}. We have $\zeta \geq (ds_{max} \cdot \delta_{min})/res^{th}$, since $res^{th} \geq ds_{min} \cdot \delta_{min}$. In the worst case, each location may have a maximum unit processing cost c_{unit}^{max}. The maximum computing resource assigned to the first virtual location of LOC_l is res^{th}. Due to Constraint (12), it is clear that the fractional assignment of each location LOC_l meets the budget, i.e., $|bs_{m,t}| \cdot x_{l,m,k}(t) \cdot c_{m,k,l,t} \leq (res^{th}/\delta_l) \cdot c_{unit}^{max} \leq B_m$.

In the integral solution, a dataset may be assigned to locations with the maximum unit processing cost. This means all the splits of a dataset in the fractional solution are moved to a single location with the maximum unit processing cost. Let us fix the first virtual location of each location. In algorithm Appro, we consider the datasets in non-increasing order of their data volume. The fractional assignment to the first virtual location usually has a higher data volume, because all virtual locations have the same δ_l. Thus, the movement of such fractional assignments to the first virtual locations incurs the maximum impact on the budget. That is, for each of the first virtual locations, the maximum data volume that can be processed by an amount res^{th} of computing resource is $\frac{res^{th}}{\delta_{min}}$. There are at most $\frac{ds_{max}\delta_{min}}{res^{th}}$ splits in each dataset assigned to the first virtual locations of other locations. In the integral solution, we thus have the worst case contribution to the processing cost by $(ds_{max}/res^{th})res^{th}c_{unit}^{max} \leq (\zeta/\delta_{min})B_m$. ■

Theorem 2: Algorithm Appro produces a feasible solution for the offline AoD minimization problem in a hierarchical MEC network G with all datasets being assigned immediately after their generation. The approximation ratio of the algorithm is $\frac{d_{max} \cdot res^{th}}{d_{min}}$, where $d_{max} = \max_{m,k,l,t}\{d_{m,k} + d_{k,l} + d_l\}$ and $d_{min} = \min_{m,k,l,t}\{d_{m,k} + d_{k,l} + d_l\}$.

Proof: We first analyze the approximation ratio. We use DA_M to denote the fractional AoD of the minimum weight maximum matching in the constructed bipartite graph G'', and OPT_{LP} to denote the optimal solution to the **LP-SC**(res^{th}). Denote by DA be the obtained integral solution from algorithm Appro. By the construction of the bipartite graph G'' and the rounding procedure, we have $DA_M = OPT_{LP}$. In algorithm Appro, it is known that the first virtual location LOC_{l_1} is assigned a single dataset, which in the worst case can be a dataset with volume res^{th} in the obtained integral solution. However, in solution OPT_{LP}, such a dataset may be fractionally assigned to many other locations. Relocating the fractional assignment in OPT_{LP} may increase the AoD of the request by at most $(d_{max} \cdot res^{th})/d_{min}$, if the first virtual location LOC_{l_1} has the lowest delay of processing a unit data, i.e., d_{min}. This means that $DA/OPT_{LP} \leq (d_{max} \cdot res^{th})/d_{min}$.

We now analyze the relationship between the parametric LP relaxation OPT_{LP} and the natural relaxation without parameter res^{th} denoted by OPT^*. Recall that we assume that $res^{th} \geq ds_{min} \cdot \delta_{min}$. We thus consider the following two cases:

Case 1: $res^{th} \geq ds_{max} \cdot \delta_{max}$. This means that the threshold of resource consumption is higher than any datasets generated by the IoT devices. res^{th} thus has no constraint on decision variable $x_{l,m,k}(t)$. Therefore, $OPT_{LP} = OPT^*$.

Case 2: $ds_{max} \cdot \delta_{max} > res^{th} \geq ds_{min} \cdot \delta_{min}$. Some datasets may not be assigned to some locations due to the violation of the threshold. This makes datasets of high resource consumptions being fractionally split into more locations, thereby reducing the AoD of the fractional assignment. We thus have $OPT_{LP} \leq OPT^*$.

Let OPT be optimal solution to the special case of the AoD minimization problem. Clearly, we have $OPT^* \leq OPT$. We have the approximation ratio $DA/OPT \leq DA/OPT_{LP} \leq (d_{max} \cdot res^{th})/d_{min}$. ■

VI. An Online Learning Algorithm for the Online AoD Minimization Problem

A. Overview

In real scenarios, processing and transmission delays are uncertain since they depend on wireless environments, congestion of links, and workloads on cloudlets and datacenters. We assume that a number of "probes" are deployed in an MEC network to monitor and predict the delays by issuing queries to network components. The predicted delays however may not be the real delays when requests are assigned to cloudlets or datacenters. We handle uncertainties of network delays via leveraging the technique of multi-armed bandits with experts [30]. Specifically, we consider the probes deployed in the MEC network as experts. These experts obtain the processing and transmission delays from base stations, cloudlets, datacenters, and links. Requests are then assigned according to the delay information, and experts observe the real delays experienced by the requests. Our basic idea is that, at the beginning of each time slot τ, each expert predicts the delay of processing and transmitting a unit amount of the dataset in the MEC network G, via an autoregressive prediction model, following the idea of the Hedge algorithm [30] with full feedback. Algorithm Appro is then invoked to assign requests to cloudlets and datacenters based on the predicted information. In the end

of time slot τ, the real delays can be obtained through the deployed probes. The cost of each expert is calculated.

B. Online Learning Algorithm

We propose the online learning algorithm. We assume that there are N experts in the MEC network G. Each pair of a base station and a cloudlet (or datacenter) is considered as an arm. In a real network deployment, an expert corresponds a probe monitoring a few network nodes and links. Each expert is in charge of a subset of arms, recommends its arms by revealing its learned delays of its arms, and receives a cost if its recommendation is different with the real delay.

Let exp_n be an expert with $1 \leq n \leq N$. Each arm corresponds two parts: a base station bs_k and a location LOC_l. Initially, the algorithm assigns each expert a weight of 1, meaning a full trust of the predicted values of its pairs of base stations and locations. As the algorithm proceeds iteratively, at each time slot τ, it degrades the weight of an expert according to the received cost (a.k.a. penalty). Specifically, the algorithm selects an expert with a probability that is proportional to its weight. Let $w_\tau(exp_n)$ be the weight of an expert exp_n and $p_\tau(exp_n)$ be the probability of choosing an expert exp_n at time slot τ. Then,

$$p_\tau(exp_n) = w_\tau(exp_n)/\big(\sum_{n'=1}^{N} w_\tau(exp_{n'})\big). \quad (22)$$

The pairs of base stations and locations (i.e., arms) of each chosen expert will be considered as candidate solutions for the requests in $R(t)$. We then invoke algorithm `Appro`, by considering the candidate solutions as its input. Algorithm `Appro` assigns the requests to cloudlets and datacenters for processing based on the delays reported by the experts. The datasets of assigned requests are then processed. Then, the real delays of processing and transmitting a unit of data are revealed. Based on the real delays, the experts receive their costs. We consider such costs as the difference between the predicted delay and the real delay. Intuitively, an expert will receive a higher cost (or penalty) if its reported delay deviates too much from the real delay. Let $d^{unit}_{m,k,l} = d_{m,k} + d_{k,l} + d_l$ be the delay of processing a dataset of an IoT device dv_m in location LOC_l via base station bs_k. Denote by $c_\tau(exp_n)$ the cost received by expert exp_n, we have

$$c_\tau(exp_n) = \sum_{\substack{bs_k \in \mathcal{BS}_n \\ LOC_l \in \mathcal{LOC}_n}} |ds_{m,t}| \cdot |d^{unit}_{m,k,l} - \hat{d}^{unit}_{m,k,l}|,$$

where $\mathcal{BS}_n$ and $\mathcal{LOC}_n$ are the sets of base stations and locations that are monitored by expert exp_n, respectively, and $\hat{d}^{unit}_{m,k,l}$ is the delay of processing $ds_{m,t}$ via bs_k in LOC_l. To avoid such experts being selected, we decrease its weight by

$$w_{\tau+1}(exp_n) = w_\tau(exp_n) \cdot (1-\epsilon)^{c_\tau(exp_n)}. \quad (23)$$

The detailed algorithm are elaborated in **Algorithm** 3, which is referred to as algorithm `OL_MAB`.

C. Regret Analysis

The rest is to analyze the regret of algorithm `OL_MAB`. We first define the regret as the expected deviation of the AoD of the obtained solution from the optimal solution, i.e.,

$$Reg(T) = DA(\texttt{OL\_MAB}) - E[DA_{exp^*}], \quad (24)$$

where $DA(\texttt{OL\_MAB})$ is the AoD obtained by `OL_MAB` and DA_{exp^*} is the AoD that is obtained based on the predictions of the best expert exp^*.

Let $cost(exp_n) = \sum_{\tau=1}^{T} c_\tau(exp_n)$ be the total cost of each expert exp_n. We then have $exp^* \in \arg\min_{1\leq n\leq N} cost(exp_n)$, and its cost is denoted by $cost^*$.

Theorem 3: Algorithm `OL_MAB` has a regret of $\frac{\ln N}{\epsilon} + 2\epsilon U_d$, assuming that there is a bound U_d on the minimum and maximum delays of transmitting and processing a unit amount of data in a pair of a base station and a cloudlet (or datacenter), i.e., $U_d = d_{max} - d_{min}$, where $d_{max} = \max_{m,k,l,t}\{d_{m,k} + d_{k,l} + d_l\}$, $d_{min} = \min_{m,k,l,t}\{d_{m,k} + d_{k,l} + d_l\}$, and ϵ is a constant with $0 < \epsilon \leq 1/2$.

Proof: Let $W_\tau = \sum_{n=1}^{N} w_\tau(exp_n)$ be the total weight of all experts before time slot τ. We observe that the weight of each expert after the last time slot of the monitoring period T can be calculated by $w_{T+1}(exp_n) = w_1(exp_n)\Pi_{\tau=1}^{T}(1-\epsilon)^{cost(exp_n)}$. It can be seen that the total weight after the monitoring period T is

$$W_{T+1} > w_{T+1}(exp^*) = (1-\epsilon)^{cost^*} > (1-\epsilon)^{DA_{exp^*}}, \quad (25)$$

due to the definition of costs and the fact that $(1-\epsilon)^\alpha > (1-\epsilon)^\beta$ for any $\alpha < \beta$.

We show the relationship between the costs of experts and the AoDs of requests.

$$\begin{aligned} W_{\tau+1}/W_\tau &= \big(\sum_{n=1}^{N} w_{\tau+1}(exp_n)\big)/W_\tau \\ &= \frac{\sum_{n=1}^{N}(1-\epsilon)^{c_\tau(exp_n)} \cdot w_\tau(exp_n)}{W_\tau}, \text{ due to Eq. (23)} \quad (26) \\ &< 1 - \alpha \sum_{n=1}^{N} p_\tau(exp_n) \sum_{\substack{bs_k \in \mathcal{BS}_n \\ LOC_l \in \mathcal{LOC}_n}} |ds_{m,t}| \cdot d^{unit}_{m,k,l} \\ &+ \beta \sum_{n=1}^{N} p_\tau(exp_n)\Big(\sum_{\substack{bs_k \in \mathcal{BS}_n \\ LOC_l \in \mathcal{LOC}_n}} |ds_{m,t}| \cdot d^{unit}_{m,k,l}\Big)^2 \\ &= 1 - \alpha E(DA(\texttt{OL\_MAB}, \tau)) + \beta E(DA(\texttt{OL\_MAB}, \tau)^2), \quad (27) \end{aligned}$$

where $DA(\texttt{OL\_MAB}, \tau)$ is the total AoD of requests scheduled at time slot τ. We then have $DA(\texttt{OL\_MAB}) = \sum_{\tau=1}^{N} DA(\texttt{OL\_MAB}, \tau)$.

By taking a logarithm on both sides of Ineq. (27), we have

$$\begin{aligned} &\ln\big(W_{\tau+1}/W_\tau\big) \\ &< \ln(1 - \alpha E(DA(\texttt{OL\_MAB}, \tau)) + \beta E(DA(\texttt{OL\_MAB}, \tau)^2)) \\ &< -\alpha E(DA(\texttt{OL\_MAB}, \tau)) + \beta E(DA(\texttt{OL\_MAB}, \tau)^2), \quad (28) \end{aligned}$$

Algorithm 3 `OL_MAB`

Input: $G = (\mathcal{BS} \cup \mathcal{LOC}, E)$, a set of requests.
Output: Request assignments.
1: Initialize the weights of experts as $w_1(exp_n) = 1$ for the expert exp_n;
2: **for** each time slot $\tau = 1 \ldots T$ **do**
3: Calculate the probability $p_\tau(exp_n)$ of selecting exp_n by Eq. (22);
4: Select each expert exp_n with probability p;
5: Each selected expert predicts the delays of its base station bs_k and location LOC_l, which are considered as candidate locations;
6: Invoke algorithm `Appro`;
7: Observe the costs of experts, and update their weights by Eq. (23);

since $\ln(1-y) < -y$ for any $y \in (0,1)$. In particular, this holds when $(\alpha, \beta) = (\epsilon, 0)$. This means

$$\begin{aligned}&\sum_{\tau=1}^{N}(\alpha E(DA(\texttt{OL\_MAB},\tau)) - \beta E(DA(\texttt{OL\_MAB},\tau)^2))\\ &< -\ln \Pi_{\tau=1}^{T}(W_{\tau+1}/W_\tau) = -\ln\left(W_{\tau+1}/W_\tau\right)\\ &= \ln W_1 - \ln W_{T+1} < \ln N - \ln(1-\epsilon) DA_{exp^*}. \qquad (29)\end{aligned}$$

With $(\alpha, \beta) = (\epsilon, 0)$, the above inequality can be written as,

$$E(DA(\texttt{OL\_MAB})) < \frac{\ln N}{\epsilon} + \frac{1}{\epsilon}\ln\frac{1}{(1-\epsilon)} E(DA_{exp^*}). \quad (30)$$

Assume $\epsilon \in (0, 1/2)$, we have $\frac{1}{\epsilon}\ln\frac{1}{(1-\epsilon)} \leq 1 + 2\epsilon$, meaning $E(DA(\texttt{OL\_MAB}) - DA_{exp^*}) < (\ln N)/\epsilon + 2\epsilon E(cost^*)$. Clearly, $cost^* < U_d \cdot ds_{max}$. The regret of OL_MAB thus is $E(DA(\texttt{OL\_MAB}) - DA_{exp^*}) < (\ln N)/\epsilon + 2\epsilon U_d$. ■

VII. Experiments

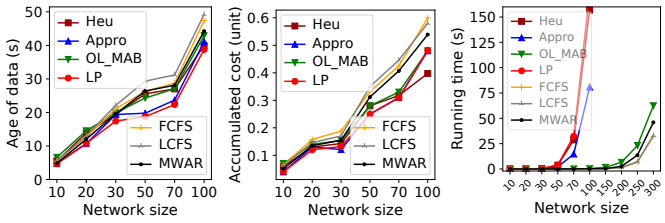
A. Parameter Settings

We consider an MEC network with 10 to 200 IoT devices, where the number of base stations, cloudlets and datacenters in the network is set to 10%, 5%, and 2% of the network size, respectively. Each IoT device is connected to at least one base station and with a 30% chance of connecting to multiple base stations. The computing capacity of each cloudlet varies from 4,000 to 12,000 MHz with several servers [36]. The computing capacity of a datacenter is randomly chosen from $\{64, 128, 256, 512\}$ GHz [16]. The network bandwidth capacity and latency of different links vary. Specifically, they are randomly drawn from the ranges of $[20, 100]$ Mbps and $[0, 5]$ milliseconds (ms) for links between base stations and IoT devices, the intervals from $[50, 2000]$ Mbps and $[5, 50]$ ms for links between base stations and cloudlets, and the ranges of $[100, 1000]$ Mbps and $[50, 1000]$ ms for links between base stations and datacenters [16], [45]. The data generation rate of each IoT devise follow typical video analysis and continuous sensor monitoring applications, which is varied from 0.01 to 100 Mbps [16]. The experiments are performed in a server with an Intel i7-9700 CPU and 64 GB memory.

We compare the performance of the proposed algorithms Heu, Appro, and OL_MAB against the following benchmarks: (1) A relaxed version of the **ILP**, represented by LP. Such comparison is conservative, because LP represents a lower bound of the optimal solution of the problem of concern; (2) the first-come-first-served (FCFS) algorithm, which queues all generated datasets in a FIFO queue on each cloudlet or datacenter; (3) the last-come-first-served (LCFS) algorithm in [3]; (4) the maximum weighted age reduction (MWAR) algorithm in [48]; (5) algorithm OL_MAB_COMP, which is the same with OL_MAB except that it has the complete information of the delays of processing and transmitting a unit amount of data; and (6) algorithm OL_MAB_NONE, which is algorithm OL_MAB without using the predicted information by experts.

B. Performance Evaluation by Simulations

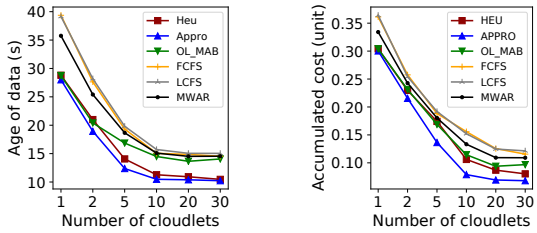
We first evaluate the performance of algorithms Heu, Appro, OL_MAB, LP, FCFS, LCFS, and MWAR in terms of the accumulated AoDs, accumulated costs, and running times by varying the network size from 10 to 100. From Fig. 3 (a), we can see that algorithms Heu and Appro have AoDs close to that of LP, implying the solutions obtained by them are near optimal. Also, algorithms Heu and Appro have 7% and 10% lower AoD than that of algorithm MWAR, respectively. The costs delivered by algorithms Heu and Appro are also close to that of LP and lower than those of the rest algorithms. The reason is that algorithm Heu adopts a fine-grained allocation strategy that divides each cloudlet into a set of virtual cloudlets, while algorithm Appro filters out the cloudlets with small fractional values of LP. Algorithm OL_MAB achieves better an accumulated AoD and a lower accumulated cost than those of algorithms FCFS, LCFS, and MWAR, respectively. The rationale behind is that the use of experts greatly improves the accuracy of delay prediction and avoids the selection of suboptimal solutions, by using the MAB method. From Fig. 3 (c), we can see that algorithm OL_MAB has less running time than those of algorithms Heu and Appro. We can also see that algorithm OL_MAB has a similar running time as those of the algorithms MWAR, LCFS, and FCFS that are widely adopted in various systems. This means that algorithm OL_MAB has the efficiency that can be adopted in real systems. The offline algorithms Heu and Appro have higher running times, because they deal with the offline problem that assumes all information for the time horizon is given. It must be mentioned that such algorithms are usually run in the network design phase, with the aim to obtain scheduling policies. Once the system starts running, the scheduler schedules requests according predetermined scheduling policies. In this sense, the running times of algorithms Heu and Appro are efficient in practice.



(a) Accumulated AoD. (b) Accumulated cost. (c) Running time.

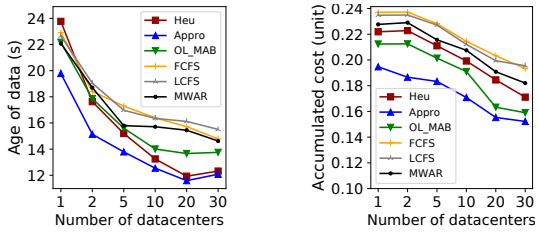
Fig. 3. The performance of algorithms **ILP**, Heu, Appro and OL_MAB.

We then evaluate the performance of algorithms Heu, Appro, OL_MAB, FCFS, LCFS, and MWAR in terms of the accumulated AoD and cost, by varying the number of cloudlets and datacenters from 1 to 30 in a network consisting of 200 IoT devices. Fig. 4 and Fig. 5 illustrate the evaluation results. From Fig. 4 (a) and (b) we can see that algorithm Appro delivers the smallest AoD and the lowest accumulated cost among the comparison algorithms. In addition, the AoDs and costs of the six mentioned algorithms decrease with the growth of the numbers of cloudlets and datacenters. This is because the total available computing resource increases with the growth of numbers of cloudlets and datacenters, and there are more opportunities that the datasets can be processed earlier and faster in locations with lower costs. Similar results of the impact of the number of datacenters are depicted in Fig. 5.



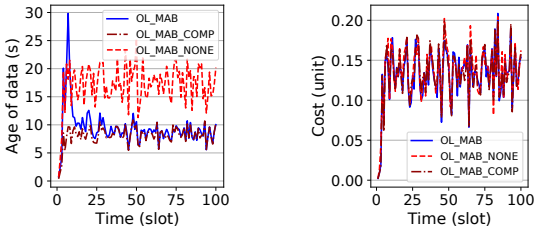
(a) Accumulated AoD. (b) Accumulated cost.

Fig. 4. The impact of the number of cloudlets on the performance of algorithms.



(a) Accumulated AoD. (b) Accumulated cost.

Fig. 5. The impact of the number of datacenters on the performance of algorithms.



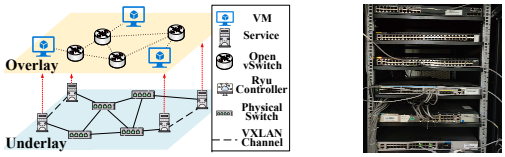
(a) AoD. (b) Cost.

Fig. 6. The convergence process of algorithm OL_MAB.

We then evaluate the convergence rate of algorithms OL_MAB, OL_MAB_COMP, and OL_MAB_NONE, in a time period of 100 time slots. As shown in Fig. 6, algorithm OL_MAB converges in 30 time slots, and has a much lower AoD and cost than those of algorithm OL_MAB_NONE. The rationale is that algorithm OL_MAB predicts the network delays accurately, by following the information provided by the experts distributed in the network, while algorithm OL_MAB_NONE does not follow the predicted delays of experts. We can also see that algorithm OL_MAB has almost the same AoD and cost as those of algorithm OL_MAB_COMP, meaning that the predictions by the experts are accurate.

C. Performance Evaluation in a Test-bed

To evaluate both the applicability in real environments and scalability of the proposed algorithms, we built a test-bed consisting of an underlay with hardware appliances and an overlay with virtual resources. The physical underlay consists of five hardware heterogeneous switches and four servers. In the overlay, by using the VXLAN technique [23], we build a virtual network with a number of Open vSwitch (OVS) [51] nodes and virtual machines, as shown in Fig. 7. We thus attach a server node in each of the switches that do not support VXLAN. We also instantiate a virtual node with built-in support for VXLAN, by replacing the network stack with OVS in Linux kernel. All the rest of settings are the same as the simulation settings.



(a) The underlay and overlay of a test bed (b) The physical deployment of hardware switches

Fig. 7. A test-bed with both hardware switches and virtual resources.

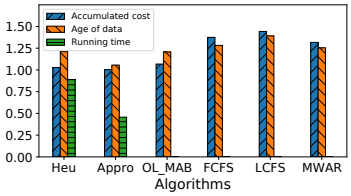


Fig. 8. The performance improvement ratio of algorithms Heu, Appro and OL_MAB against LP in a test-bed.

We evaluate the performance of algorithms Heu, Appro, OL_MAB, FCFS, LCFS, and MWAR in the built test-bed, in terms of the ratios of obtained the AoD and cost over those of LP. Fig. 8 shows the evaluation results. We can see that algorithm OL_MAB delivers 4% lower AoD and 20% lower cost than those delivered by algorithm MWAR. Also, algorithm Appro has the lowest AoD while algorithm LCFS has the highest AoD, because algorithm Appro schedules each request immediately after its arrival while algorithm LCFS schedules the last arrived request.

VIII. Conclusion

In this paper, we investigated the AoD minimization problem for IoT big data processing in a hierarchical MEC network. We first formulated an ILP solution for the problem. For the offline AoD minimization problem, we devised an efficient heuristic. We also proposed an approximation algorithm with a provable approximation ratio for a special case of the problem. For the online AoD minimization problem, we developed an online learning algorithm with a bounded regret, based on a multi-armed bandit method. We finally evaluated the performance of the proposed algorithms by extensive simulations and implementations in a real test-bed. Results show that the proposed algorithms outperform existing studies by reducing the AoD around 10%.

Acknowledgements

The work by Zichuan Xu and Qiufen Xia is funded by the National Natural Science Foundation of China (NSFC) (Grant No. 62172068, 62172071, 61802048, and 61802047), and the "Xinghai scholar" program. The work by Weifa Liang is supported by the Australian Research Council Discovery Project (Grant No. DP200101985), and City University of Hong Kong (Grant No. 9380137/CS). The work done by Wenzheng Xu is supported in part by NSFC (Grant No. 61602330). The work done by Pan Zhou is supported in part by NSFC (Grant No. 61972448).

REFERENCES

[1] S. Arora, and B. Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[2] G. S. Aujla, N. Kumar, A. Y. Zomaya, and R. Ranjan. Optimal decision making for big data processing at edge-cloud environment: an SDN perspective. *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 2, pp. 778–789, IEEE, 2017.

[3] A. M. Bedewy, Y. Sun, and N. B. Shroff. Minimizing the age of information through queues. *IEEE Transactions on Information Theory*, Vol.65, No.8, pp.5215–5232, 2019.

[4] Y. Cheng, and G. Xu. A novel task provisioning approach fusing reinforcement learning for big data. *IEEE Access*, Vol. 7, pp. 143699–143709, IEEE, 2019.

[5] J. P. Champati, R. R. Avula, T. J. Oechtering, and J. Gross. On the minimum achievable age of information for general service-time distributions. *Proc. of INFOCOM*, IEEE, 2020.

[6] M. B. Cohen, Y. T. Lee, and Z. Song. Solving linear programs in the current matrix multiplication time. *Proc. of STOC*, ACM, 2019.

[7] L. Corneo, C. Rohner, and P. Gunningberg. Age of information-aware scheduling for timely and scalable internet of things applications. *Proc. of INFOCOM*, IEEE, 2019.

[8] Q. Fan and N. Ansari. Cost Aware cloudlet placement for big data processing at the edge. *Proc. of ICC*, IEEE, 2017.

[9] D. Golomb, D. Gangadharan, S. Chen, *et al.* Data freshness over-engineering: formulation and results. *Proc. of ISORC*, IEEE, 2018.

[10] Google Cloud Dataflow. https://cloud.google.com/dataflow, Accessed in June 2021.

[11] Enhanced data freshness, Google Analytics. https://support.google.com/analytics/answer/7084038, Accessed in June 2021.

[12] Q. He, G. Dán, and V. Fodor. Joint assignment and scheduling for minimizing age of correlated information. *IEEE/ACM Transactions on Networking*, Vol.27, No.5, pp.1887–1900, 2019.

[13] B. Heintz, A. Chandra, and R. K. Sitaraman. Trading timeliness and accuracy in geo-distributed streaming analytics. *Proc. of SoCC*, ACM, 2016.

[14] L. Huang, S. Bi, and Y. J. Zhang. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, IEEE, 2019.

[15] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis. The cost of delay in status updates and their value: Non-linear ageing. To appear in *IEEE Transactions on Communications*, 2020.

[16] Q. Kuang, J. Gong, X. Chen, and X. Ma. Age-of-information for computation-intensive messages in mobile edge computing. *Proc. of WCSP*, IEEE, 2019.

[17] R. Li and H. Asaeda. MWBS: An efficient many-to-many wireless big data delivery scheme. *IEEE Transactions on Big Data*, Vol. 6, No. 2, pp. 233–247, IEEE, 2020.

[18] C. Li, S. Li, Y. Chen, Y. T. Hou, and W. Lou. AoI scheduling with maximum thresholds. *Proc. of INFOCOM*, IEEE, 2020.

[19] H. Li, H. Xu, and S. Nutanong. Bohr: similarity aware geo-distributed data analytics. *Open Access Media*, USENIX, 2017.

[20] J. Lou, X. Yuan, S. Kompella, and N. F. Tzeng. AoI and throughput tradeoffs in touting-aware multi-hop wireless networks. *Proc. of INFOCOM*, IEEE, 2020.

[21] N. Lu, B. Ji, and B. Li. Age-based scheduling: Improving data freshness for wireless real-time traffic. *Proc. of MobiHoc*, ACM, 2018.

[22] Z. Lu, N. Wang, J. Wu, and M. Qiu. IoTDeM: an IoT big data-oriented MapReduce performance prediction extended model in multiple edge clouds. *Journal of Parallel and Distributed Computing*, Vol. 118, No. 2, pp. 316–327, Elsevier, 2018.

[23] M. Mahalingam et al. Virtual extensible local area network (VXLAN): a framework for overlaying virtualized layer 2 networks over layer 3 networks. RFC 7348, IETF, *https://tools.ietf.org/html/rfc7348*.

[24] S. Misra, A. Mondal, and S. Khajjayam. Dynamic big-data broadcast in fat-tree data center networks with mobile IoT devices. *IEEE Systems Journal*, Vol.13, No.3, pp.2898–2905, IEEE, 2019.

[25] Multi-access Edge Computing (ETSI). ttps://www.etsi.org/tecnologies/multi-access-edge-computing.

[26] H. Pang, J. Liu, X. Fan, and L. Sun. Toward smart and cooperative edge caching for 5g networks: a deep learning based approach. *Proc. of IWQoS*, IEEE, 2018.

[27] H. Pang, C. Zhang, F. Wang, *et al.* Optimizing personalized interaction experience in crowd-interactive livecast: a cloud-edge approach. *Proc. of ACM Multimedia*, 2018.

[28] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica. Low latency analytics of geo-distributed data in the wide area. *Proc. of SIGCOMM*, ACM, 2015.

[29] Z. Qian, F. Wu, J. Pan, K. Srinivasan, and N. B. Shroff. Minimizing age of information in multi-channel time-sensitive information update systems. *Proc. of INFOCOM*, IEEE, 2020.

[30] A. Slivkins. Introduction to multi-armed bandits. 2019. [Online]. Available: http://arxiv.org/abs/1904.07272

[31] X. Song, X. Qin, Y. Tao, B. Liu, and P. Zhang. Age based task scheduling and computation offloading in mobile-edge computing systems. *Proc. of WCNCW*, IEEE, 2019.

[32] Y. Sun and B. Cyr. Sampling for data freshness optimization: non-linear age functions. *Journal of Communications and Networks*, Vol. 21, No. 3, pp. 204–219, IEEE, 2019.

[33] R. Talak, S. Karaman, and E. Modiano. Optimizing information freshness in wireless networks under general interference constraints. *IEEE/ACM Transactions on Networking*, Vol. 28, No. 1, pp. 15–28, IEEE, 2019.

[34] R. Talak, S. Karaman, and E. Modiano. Improving age of information in wireless networks with perfect channel state information. *IEEE/ACM Transactions on Networking*, DOI: 10.1109/TNET.2020.2996237, 2020.

[35] C. H. Tsai, and C. C. Wang. Unifying AoI minimization and remote estimation-optimal sensor/controller coordination with random two-way delay. *Proc. of INFOCOM*, IEEE, 2020.

[36] S. Wan, J. Lu, P. Fan, and K. B. Letaief. Towards big data processing in IoT: path planning and resource management of UAV base stations in mobile-edge computing system. *IEEE Internet of Things Journal*, Vol. 7, No. 7, pp. 5995–6009, 2019.

[37] M. Wang, W. Chen, and A. Ephremides. Real-time reconstruction of a counting process through first-come-first-serve queue systems. *IEEE Transactions on Information Theory*, Vol. 66, No. 7, pp. 4547–4562, 2020.

[38] K. H. Wong, J. Cao, Y. Yang, *et al.* BigARM: A big-data-driven airport resource management engine and application tools. *Proc. of DASFAA*, Springer, 2020.

[39] Q. Xia, L. Bai, W. Liang, Z. Xu, L. Yao, and L. Wang. QoS-aware proactive data replication for big data analytics in edge clouds. *Proc. of ICPP*, New York, IEEE, 2019.

[40] Q. Xia, W. Ren, and M. Li. Age-aware query evaluation for big data analytics in mobile edge clouds. *Proc. of HPCC*, Fiji, IEEE, 2020.

[41] Q. Xia, Z. Xu, W. Liang, S. Yu, S. Guo, and A. Y. Zomaya. Efficient data placement and replication for QoS-aware approximate query evaluation of big data analytics. *IEEE Transactions on Parallel and Distributed Systems*, Vol.20, No.12, pp.2677–2691, IEEE, 2019.

[42] Q. Xia, Z. Xu, W. Liang, and A. Y. Zomaya. Collaboration- and fairness-aware big data management in distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, Vol.27, No.7, pp.1941–1953, IEEE, 2015.

[43] Q. Xia, L. Zhou, W. Ren, and Y. Wang. Proactive and intelligent evaluation of big data queries in edge clouds with materialized views. *Computer Networks*, Elsevier, 2021.

[44] R. D. Yates. The age of information in networks: Moments, distributions, and sampling. *IEEE Transactions on Information Theory*, DOI: 10.1109/TIT.2020.2998100, 2020.

[45] R. Yu, G. Xue, and X. Zhang. Provisioning QoS-aware and robust applications in internet of things: a network perspective. *IEEE/ACM Transactions on Networking*, Vol.27, No.5, pp.1931–1944, IEEE, 2019.

[46] L. Zhang, A. Sun, R. Shea, J. Liu, and M. Zhang. Rendering multi-party mobile augmented reality from edge. *Proc. of NOSSDAV*, ACM, 2019.

[47] Z. Zhuang, J. Wang, Q. Qi, J. Liao and Z. Han. Adaptive and robust network routing based on deep reinforcement learning with lyapunov optimization. *Proc. of IWQoS*, IEEE, 2020.

[48] J. Zhong, W. Zhang, R. D. Yates, A. Garnaev, and Y. Zhang. Age-aware scheduling for asynchronous arriving jobs in edge applications. *Proc. of INFOCOM*, IEEE, 2019.

[49] J. Zhong, R. D. Yates, and E. Soljanin. Multicast with prioritized delivery: how fresh is your data? *Proc. of SPAWC*, IEEE, 2018.

[50] Z. Zhou and L. Zhao. Cloud computing model for big data processing and performance optimization of multimedia communication. *Journal of Computer Communications*, Elsevier, 2020.

[51] Open vSwitch. *https://www.openvswitch.org*