Online Learning Algorithms for Offloading Augmented Reality Requests with Uncertain Demands in MECs

Zichuan Xu[†], Dongqi Liu[†], Weifa Liang[‡], Wenzheng Xu[§*], Haipeng Dai[¶], Qiufen Xia[$], and Pan Zhou[%]
[†] School of Software, Dalian University of Technology, Dalian, China, 116621.
[‡] Department of Computer Science, City University of Hong Kong, Hong Kong.
[$] International School of Information Science and Engineering, Dalian University of Technology, Dalian, China, 116621.
[§] College of Computer Science, Sichuan University, Chengdu, Sichuan, China
[¶] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[%] The Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, China
Emails: z.xu@dlut.edu.cn, ldq0913@mail.dlut.edu.cn, weifa.liang@cityu.edu.hk
wenzheng.xu@scu.edu.cn, haipengdai@nju.edu.cn, qiufenxia@dlut.edu.cn, panzhou@hust.edu.cn.
[*] Corresponding author.

***Abstract*—Augmented Reality (AR) has various practical applications in healthcare, education, and entertainment. To provide a fully interactive and immersive experience, AR applications require extremely high responsiveness and ultra-low processing latency. Mobile edge computing (MEC) has shown great potential in meeting such stringent requirements and demands of AR applications by implementing AR requests in edge servers within the close proximity of these applications.**

In this paper, we investigate the problem of reward maximization for AR applications with uncertain demands in an MEC network, such that the reward of provisioning services for AR applications is maximized and the responsiveness of AR applications is enhanced, subject to both network resource capacity. We devise an exact solution for the problem if the problem size is small, otherwise we develop an efficient approximation algorithm with a provable approximation ratio for the problem. We also devise an online learning algorithm with a bounded regret for the dynamic reward maximization problem without the knowledge of the future arrivals of AR requests, by adopting the technique of Multi-Armed Bandits (MAB). We evaluate the performance of the proposed algorithms through simulations. Experimental results show that the proposed algorithms outperform existing studies by 17% higher reward.

***Index Terms*—Mobile edge computing, augmented reality, reward maximization, online learning, Multi-Armed Bandits (MAB).**

I. Introduction

Augmented Reality (AR) that inserts virtual content into a stream of views of the real world captured from the physical environment by various sensors and cameras is emerging as a popular application in many application areas [15], [12]. On the start of an AR application, it starts to capture the physical environment and sends video streams to the mobile edge computing (MEC) network at a certain data rate. One key requirement of AR applications is to handle their data streams timely; otherwise, the insertion of virtual content can cause a significant delay between the observations of the world and the moment when the AR display is presented to users [18]. With the fast development of 5G communications, MEC is envisioned as the key enabling technology that brings content and computing resources close to AR users, reducing latency and workload on backhaul networks [16], [18]. As such, various service providers are seeking opportunities of promoting their revenues by deploying AR services with strong real-time guarantees in the edge of core networks. For example, the Verizon team recently built an independent GPU-based system for AR applications, which shows potential in paving the way for a new class of affordable AR services [35]. In particular, service providers can place their AR services to locations in an MEC network and offload user tasks to the placed AR services for execution, which is referred to as *task offloading*.

Task offloading for AR applications poses several crucial challenges. First, network service providers obtain rewards by offloading AR requests to an MEC network. Quantifying such rewards is important for network service providers to guide their strategy of adopting MEC. This however is very challenging. The network service provider usually cannot determine the reward of offloading an AR request in an MEC network solely based on the rates of data streams of AR applications. The reason is that the reward depends on many other factors such as resource usage costs, processing delays, and pricing models of the network service provider [8]. Also, the actual rate of each data stream of an AR application is normally not known in advance before the admission of the AR request. How to cache AR services and offload tasks based on such uncertain data rates and rewards is challenging. Second, a deterministic relationship between reward and data rates, such as a proportional model, is not realistic. Instead, the rewards and data rates of requests are independent. How to maximize the reward that is independent of the demand of requests is the second challenge. Third, along with the responsiveness requirement of each AR application, the continuous processing of its data stream after its being responded needs to be performed within a specified delay requirement. Considering the afore-mentioned uncertainty of data rates and rewards, how to jointly improve the responsiveness of AR applications in an MEC network while meeting their delay requirements is challenging. To address these challenges, we investigate the problem of reward maximization for AR applications in an MEC network, with the aim to enhance the responsiveness of AR applications via offloading AR requests to base stations of the MEC network. We consider uncertainties of reward and data rates and demand-independent rewards.

There are studies of task offloading and service placement in MECs under conventional task models [3], [10], [11], [17], [20], [21], [23], [29], [30], [31], [32], [33], and proportional reward function models [1], [14]. Some of them even ignored the responsiveness of AR applications [5], [19], [26], [27]. To the best of our knowledge, this is the first study that considers uncertain data rates, rewards, and demand-independent rewards while offloading AR requests in an MEC network.

The main contributions of this paper are as follows.

- We propose exact and approximation algorithms for the reward maximization problem with a set of non-preemptive AR requests, by assuming that the tasks of each AR request can be consolidated into a single base station of the MEC network for processing.
- We devise an efficient heuristic for the reward maximization problem with a set of non-preemptive AR requests, when the tasks of each AR request need to be distributed into multiple base stations.
- In real scenarios AR requests arrive into the system dynamically and can be preempted for later execution, we thus devise an online learning algorithm with a bounded regret for the dynamic reward maximization problem for a set of preemptive AR requests, to jointly optimize the waiting time before scheduling and the processing delay for implementing each AR request.
- We evaluate the performance of the proposed algorithms by simulations. Experimental results show that our algorithms outperform their counterparts by at least 17% higher reward.

The organization of this paper is as follows. Section II introduces recent studies on AR task offloading. Section III describes the system model and defines the problems. Section IV proposes the exact and approximation algorithms for the reward maximization problem with a set of non-preemptive AR requests. Section V devises the online learning algorithm for the dynamic reward maximization problem. Section VI evaluates the performance of the algorithms, and Section VII concludes the paper.

II. Related Work

Task offloading and service placement in MEC networks are explored to improve the latency of mobile services [3], [10], [11], [17], [20], [23], [21], [32], [33]. Most of these studies focused on the conventional request model, assuming that each request is indivisible, which cannot be applied to AR requests with a sequence of tasks. For example, Bi *et al.* [3] focused on the offloading of AR requests with the aim of maximizing computation rate of edge devices. Esraghi *et al.* [10] investigated a problem of joint task offloading and resource allocation in MEC networks, to minimize a weighted sum of average cost. Farhadi *et al.* [11] studied a problem of joint service placement and request scheduling, with an aim to maximize the expected number of requests served per time slot. Jošilo *et al.* [17] investigated a computation offloading problem for a set of selfish mobile devices to minimize the completion time of offloaded tasks, by formulating the interaction between mobile devices and operator as a Stackelberg game. Liu *et al.* [20] formulated a job response time minimization problem in a cluster of heterogeneous servers with the constraint of computing resource and applied online convex optimization technique to design an online scheduling algorithm with resource packing.

There are studies on offloading requests with each having a sequence of dependent tasks [23], [21], [32], [34]. Most of them however assumed that the resource demands of requests are given in advance. Considering that AR applications send their tasks in the form of data streams, the data rates and total resource demand are unknown. The results of these studies are unlikely to be applicable to the task offloading of AR applications. For example, Ren *et al.* [23] formulated a task latency minimization problem, by proposing an algorithm to not only schedule task offloading requests but also perform the allocations of communication and computation resources. Ma *et al.* [21] focused on the cooperative service placement and workload scheduling in MEC networks with an aim to minimize the response time by designing a heuristic based on Gibbs sampling. Zhao *et al.* [34] investigated the problem of offloading dependent tasks with service placement with an objective to minimize the make-span of offloaded tasks.

There are also studies on maximizing providers' reward in either MEC networks or cloud computing environments [1], [9], [14]. Nevertheless, the reward functions in these studies are assumed to be proportional to the demand of services. For example, Hadji *et al.* [14] discussed the profit maximization problem in a federated cloud environment, to balance the workload between members in the alliance. Badshan *et al.* [1] proposed efficient algorithms to optimization prices, scheduling and migration decision of virtual machines in a cloud computing environment, aiming at maximizing the profit of providers by customer satisfaction and efficient resource provision.

Meanwhile, the problem of task offloading for AR applications had gained a few attention [5], [19], [26], [27]. None of them however investigated the optimization of responsiveness of AR applications, nor considered network uncertainties of MEC networks. Bohez *et al.* [4] is one of the earliest studies to offload AR tasks in MEC networks. They developed a platform that is capable of autonomously deploying software components to minimize average CPU load, while guaranteeing smooth collaboration. For instance, Liu *et al.* [19] considered the resource allocation problem for a multi-user AR system in an MEC network with the aim to minimize the latency of users, by allowing partial task offloading and data sharing. In addition, a collaborative service placement framework for applications with adaptive bitrate-videos was proposed in [26], with the objective to minimize the expected delay of video retrieval. Wang *et al.* [27] studied the problem of dynamic configuration adaptations for AR applications in MEC networks, to minimize the per frame energy consumption of AR applications. Braud [5] investigated a problem of offloading AR tasks to mobile devices, edge servers and remote clouds via multiple offloading paths, with the aim to minimize system latency. Although Chang *et al.* [6] considered network uncertainties in the problem of task offloading in MEC networks, they considered a single edge node, and ignored the backhaul wired bandwidth consumption.

III. System Model and Problem Definitions

In this section, we first describe the system model, notions and notations. We then define the problem precisely.

A. System Model

We consider an MEC network, denoted by $G = (\mathcal{BS}, E)$, where $\mathcal{BS}$ is a set of 5G base stations, and E is the set of paths that interconnect the base stations in the backhaul of the MEC network G. Each base station $bs_i \in \mathcal{BS}$ is equipped with a certain amount of computing resource, such as a neural network accelerator, to process AR requests. Let $C(bs_i)$ be the computing capacity on bs_i. Denote by $e \in E$ a path that interconnects two base stations in the backhaul of G.

An AR application runs within a user device, and sends inference requests to the MEC network for processing continuously. Without loss of generality, we consider that each AR application sends its inference requests via its closest base station. Due to the limited computing capacity on each base station bs_i, it may not be able to implement all inference requests. We thus consider that the inference requests of each AR application can be distributed to multiple base stations in $\mathcal{BS}$ for processing, via the backhaul paths of G. Fig. 1 shows an example of the MEC network G.

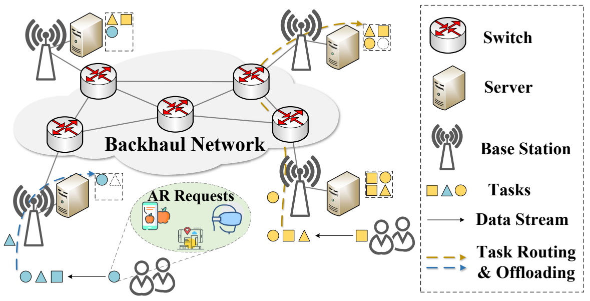


Fig. 1. An example of the MEC network

B. AR Requests and Services with Uncertain Data Rates

We consider web AR applications, which normally are supported by context-aware web browsers. Such a web AR application analyzes the surroundings of a user and combines geographic locations and temporal information with images captured by the application, such that specific locations are displayed in web browsers in AR. The video stream is fed into an AR processing pipeline to render virtual objects into the original video stream after a series of intermediate processing. Following many existing studies [5], [19], [26], [27], we divide an AR processing pipeline into a sequence tasks. A typical sequence of tasks of each AR processing pipeline can include pose estimation, mapping, world model creation, and rendering. It is known that the rendering processing is the most computing-intensive task of each AR processing pipeline. Each task takes the output matrix of its predecessor as its input matrix and produces its output matrix as its successor task. Denote by r_j a request of an AR application, and let $\{M_{j,1}, \cdots, M_{j,k}, \cdots M_{j,K_j}\}$ be the sequence of tasks of r_j, where K_j is a positive integer and $1 \le k \le K_j$.

The video of an AR application is streamed to its tasks for processing. Let ρ_j be the data rate of video streams, and such a video stream needs to be processed by the tasks of an AR processing pipeline. In fact, such data rates of an AR request depend on many factors, including the network condition, physical environment, data sampling methods of user devices, etc. Such information thus cannot be obtained in advance. However, historical information about such data rates can be obtained. Since we consider one type of service, i.e., AR service, we assume that the data rates of AR requests follow probability distributions over a set of possible data rates. Let $\mathcal{DR}$ be a finite set of possible data rates that each AR request may have. The values in set $\mathcal{DR}$ can be obtained from historical information of AR applications. The actual data rate of each AR request at a specific time thus is a value in set $\mathcal{DR}$. Processing the data of each request consumes computing resource. Let C_{unit} be the amount of computing resource that is needed to process a unit data rate of a request.

AR tasks need to be executed by AR services that are placed in the MEC network. Denote by S_j the service required by request r_j. An *instance* of each S_j consists of the trained models and meta-data that are required to execute the tasks of r_j. In conventional AR applications, such services are instantiated in remote data centers from their users, thereby increasing the delays of executing user tasks. In MEC networks, such service instances can be placed to base stations of the network within the proximity of users. Considering that each instance of S_j is usually implemented in a Virtual Machine (VM) of the MEC network. We assume that the computing resource assigned to each VM is given as a priori; for example, each VM may be assigned a CPU core and a Neural Network (NN) accelerator. Without loss of generality, we further assume that an instance of S_j is created for the execution of each request r_j and destroyed after its departure.

C. Rewards of Processing AR Requests

As the service provider of an MEC network, it receives a reward for the implementation of each AR request. Without loss of generality, we assume that the reward of implementing request r_j is correlated to its data rate ρ_j. The correlation of data rates and rewards however may be hard to obtain. The reason is that different service providers may have different pricing and cost models, such as resource usage costs and electricity costs. Even for a single service provider that operates an MEC network, its rewards of implementing requests with the same data rate vary in different time periods. We thus assume that, for each AR request r_j, there is a probability distribution over pairs of $(data\ rate,\ reward)$. That is, for each r_j and each data rate $\rho \in \mathcal{DR}$, we have $(\pi_{j,\rho}, RD_{j,\rho})$ denoting that the probability of AR request r_j with a data rate ρ is $\pi_{j,\rho}$, and the corresponding reward is $RD_{j,\rho}$.

D. Response Latency of AR Services

The processing of AR requests has various delays. AR requests arrive into the system and wait for being responded. The delay incurred during the waiting for receiving its first response is referred to as *the response delay*. Assuming that time is equally divided into time slots, each request r_j arrives into the system at a specific time slot. Denoted by a_j the arrival time of request r_j. The system then decides when to schedule request r_j. Let b_j be the time slot that the first frame of r_j's video stream is processed. The delay of waiting for scheduling request r_j thus is $b_j - a_j$.

Once a request r_j is scheduled, its video stream will be forwarded to an instance of its service S_j for processing. The data transmission to the service instance incurs the transmission delay. Video processing by the service instance leads to a processing delay. The data of each request r_j is video frames. The service of each request needs to process the video frames by recognizing objects and rendering the augmented data to the original video frames. The delay that affects the user's experiences of using AR applications thus depends on how quickly each augmentation is added into each video frame. Denote by ρ_{unit} the minimum size of video frames that is needed to perform each augmentation. We assume that ρ_{unit} is given and identical for all requests. The delays of processing ρ_{unit} in different base stations varies. Let d^{pro}_{jki} be the delay of processing ρ_{unit} amount of data by task $M_{j,k}$ of r_j in base station bs_i. Similarly, denote by d^{trans}_{je} the delay of transmitting ρ_{unit} amount of data along link e in the MEC network G.

Let D_j be the experienced latency of AR request r_j, which includes the delay of waiting for being scheduled, transmission and processing latencies. Each AR request has a requirement on its experienced latency. Let $\hat{D}_j$ be such a latency requirement. The experienced latency of each request r_j must no greater than its specified requirement $\hat{D}_j$, i.e.,

$$D_j \leq \hat{D}_j. \tag{1}$$

E. Problem Definitions

Given an MEC network $G = (\mathcal{BS}, E)$ with a set $\mathcal{BS}$ of base stations and a set R of requests of web AR applications, the tasks of each AR request $r_j \in R$ needs to be executed in instances of service S_j deployed in G. Specifically, we consider the following optimization problems.

The reward maximization problem with a set of non-preemptive AR requests is to maximize the expected reward of implementing the AR requests in R, by placing the instances of services to base stations in $\mathcal{BS}$ and assigning the tasks of each request to its placed service instances, subject to the computing capacity on each base station and the latency requirement of each request.

Assuming that the task execution can be preempted and resumed for execution later, *the dynamic reward maximization problem* for a given monitoring period T is to maximize the expected reward of all admitted requests for the time period T, by placing the instances of services to base stations in $\mathcal{BS}$ and scheduling the tasks of each request to its placed service instances, subject to the computing capacity on each base station and the latency requirement of each AR request.

IV. Approximation Algorithm for the Reward Maximization Problem

We now devise efficient algorithms for the reward maximization problem with a set R of non-preemptive AR requests. We first assume that all tasks of each request $r_j \in R$ are consolidated into a single base station and propose an approximation algorithm. We then extend the proposed approximation algorithm to solve the problem without the assumption.

A. Approximation Algorithm for a Special Case of the Problem

Assuming that all the tasks of each request r_j are consolidated into a single base station, the basic idea of our algorithm is to first propose an exact solution to problem by formulating an Integer Linear Program (ILP) to the problem. We then perform a novel relaxation of the ILP, based on which we design a randomized algorithm for the problem.

Exact solution: We use x_{ji} to denote whether the tasks of request r_j are assigned to base station bs_i for execution, assuming that its required service S_j is cached in bs_i. The latency D_j of each AR request r_j can be calculated by

$$D_j = b_j - a_j + \sum_{bs_i \in \mathcal{BS}} x_{ji} \big(\sum_{e \in p_{ji}} 2 \cdot d^{trans}_{je} + \sum_k d^{pro}_{jki} \big), \tag{2}$$

where p_{ji} is the shortest path between the mobile user of r_j and bs_i in the MEC network G in terms of transmission latency.

Recall that the objective of the reward maximization problem with a set of non-preemptive AR requests is to maximize the expected reward of admitted AR requests. Denote by $E(\rho_j)$ the expected size of AR request r_j. We thus can formulate the special case of the problem with tasks being consolidated into a single base station in the following.

$$\textbf{ILP-RM:} \quad \max \sum_{j,i} x_{ji} \cdot \pi_{j,\rho} \cdot RD_{j,\rho},$$

subject to,

$$\sum_i x_{ji} \leq 1, \forall r_j \tag{3}$$

$$\sum_j x_{ji} \cdot E(\rho_j) \cdot C_{unit} \leq C(bs_i), \text{ for } bs_i \in \mathcal{BS} \tag{4}$$

$$D_j \leq \hat{D}_j, \text{ for each } r_j \tag{5}$$

$$x_{ji} \in \{0, 1\}, \tag{6}$$

An LP relaxation: A simple relaxation of the **ILP-RM** is to consider x_{ji} as a real value in the range of $[0, 1]$. The obtained fractional solution has a fractional value for each x_{ji}, representing the probability of assigning request r_j to a base station bs_i. In the deterministic version of the problem solution, this may be a promising solution. On the contrary, the objective of our reward maximization problem is to maximize the expected reward of admitted requests. If we follow the mentioned simple relaxation, each request may instantiate to a higher data rate to gain a higher expected reward. This creates contention of each base station that prefers to admit requests with larger data rates. However, the probability of requests with large data rates is usually small [10]. The expected reward obtained thus may be sub-optimal.

We observe that a computing resource capacity $C(bs_i)$ is enforced for each bs_i. To tackle the contention of requests with large data sets, our basic idea is to adopt a *resource-slot-indexed* relaxation of the **ILP-RM**. Specifically, we partition the capacity $C(bs_i)$ of base station bs_i into the same amount of resource at *each resource slot*, where a resource slot represents a portion of the computing capacity of a base station. Denote by C_l be the capacity of resource slot l. Each the resource of bs_i thus is divided into

$$L = \lfloor C(bs_i)/C_l \rfloor$$

resource slots. The resource slots of each base station are indexed by l, and assigned to requests one by one according to the index. Namely, if request r_j is assigned a *starting resource slot* l, it means that the computing resource assigned to r_j starts with resource slot l of bs_i. It however may span multiple resource slots of bs_i starting from the lth resource slot. The reason is that the amount of computing resource allocated at resource slot l may not be sufficient to implement request r_j, considering that the data rate of r_j is not known in advance. Fig. 2 shows an example of a slotted base station.

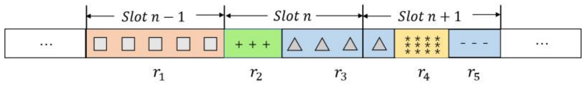


Fig. 2. Requests placed in different computing resource slots of bs_i

We now describe the proposed LP relaxation of **ILP-RM**. Variable y_{jil} is a binary indicator variable that shows whether request r_j is assigned to the lth resource slot of base station bs_i. Clearly, we have

$$x_{ji} = \sum_{l} y_{jil}. \tag{7}$$

Denote by ER_{jil} the expected reward if $y_{jil} = 1$. We have

$$ER_{jil} = \sum_{\rho \in \mathcal{DR}: \rho \cdot C_{unit} \leq C(bs_i) - l \cdot C_l} \pi_{j,\rho} \cdot RD_{j,\rho}. \tag{8}$$

Eq. (8) captures the fact that no reward is obtained if the rest resource slots (i.e., the ones after the lth resource slot) do not have enough computing resource to process the actual data rate.

The relaxed LP, denoted by **LP**, then can be written as

$$\textbf{LP:} \quad \max \sum_{jil} y_{jil} \cdot ER_{jil},$$

subject to

$$\sum_{il} y_{jil} \leq 1, \forall r_j \tag{9}$$

$$\sum_{jil': l' \leq l} y_{jil'} \cdot E(\min\{\rho_j, \frac{l \cdot C_l}{C_{unit}}\}) \leq 2 \cdot \frac{l \cdot C_l}{C_{unit}}, \quad \forall l \tag{10}$$

$$D_j \leq \hat{D}_j, \ \forall r_j \tag{11}$$

$$0 \leq y_{jil} \leq 1, \tag{12}$$

where Constraint (9) indicates that each request can only be assigned to start at a resource slot of a base station. Constraint (10) includes a truncated random variable $y_{jil'}$ to pose a limitation on the data rates of requests that can fit into the first $2 \cdot l$ resource slots. This can avoid the contention of assigning requests that have large data rates with very low probabilities to the first $2 \cdot l$ resource slots. Constraint (11) is the delay requirement of the problem, where D_j is calculated by substituting x_{ji} in Eq. (2) with $\sum_l y_{jil}$. Constraint 12 says that y_{jil} is a real value in the range of $[0, 1]$.

Approximation algorithm: We now propose an approximation algorithm for the problem. The basic idea behind the algorithm is to first obtain a fractional solution y to the **LP**. We then assign request r_j to resource slot l of base station bs_i with probability $\frac{y_{jil}}{4}$, and completely ignore r_j with probability $1 - \frac{y_{jil}}{4}$. We then add the requests to the base stations slot-by-slot. Specifically, for resource slot l of each base station bs_i, there may be multiple requests that are randomly assigned to it. Recall that the scheduling algorithm does not know the data rates of each request. However, after the scheduling of each request, it may instantiate its data rate and reveal the information to the system. In the next stage of the algorithm, we determine the assignment of the randomly assigned requests according to the revealed data rate information of currently executing requests in the system. Specifically, denote by $R_{l,i}$ the number of requests that are randomly assigned to resource slot l of bs_i. Considering a request r_j with the smallest data rate, we assign it to bs_i for execution if and only if the requests that are already assigned to bs_i occupy no more than $l \cdot C_l$ amount of computing resource. The detailed steps are given in algorithm 1, referred to as algorithm `Appro`.

B. An Efficient Heuristic for the Reward Maximization Problem

So far we assumed that the tasks of each request are consolidated into a single base station. This however may lead to the rejections of some requests due to the lack of computing resource, as shown in Step 6 in algorithm `Appro`. We here remove this assumption and propose an efficient heuristic.

Our basic idea to perform a 'pre-assignment' according to algorithm `Appro`. The obtained solution will be considered as the final assignment if there are no requests being rejected due to Step 6. Otherwise, we adjust the 'pre-assignment' by avoiding the migrations of some tasks belong to the already

Algorithm 1 An approximation algorithm for the reward maximization problem with tasks of each request are consolidated into a single base station, i.e., `Appro`

```
Input: An MEC network $G = (\mathcal{BS}, E)$, a set of requests with each
    request $r_j$ having its tasks being consolidated into a single base
    station, the data rate of each request is not known in advance
    until it is scheduled.
Output: An offloading decision for each request $r_j$.
1: Obtain a fraction solution $y$ by solving the LP;
2: Assign request $r_j$ to resource slot $l$ of base station $bs_i$ with
   probability $\frac{y_{jil}}{4}$, and completely ignore $r_j$ with probability $1 - \frac{y_{jil}}{4}$;
3: for $l \leftarrow 1, \cdots, L$ do
4:     for each base station $bs_i \in \mathcal{BS}$ do
5:         Consider the request with the $l$th smallest data rate;
6:         if the requests assigned so far in $bs_i$ occupy at most $l \cdot C_l$
           amount of computing resource then
7:             Assign $r_j$ to base station $bs_i$;
```

pre-assigned requests to nearby base stations, so that the latency requirement is not violated and the to-be-assigned request is admitted (condition in step 6 is met). If there is no such pre-assigned requests that can be migrated, we reject request r_j. Specifically, let R^{Pre} be the set of requests that are pre-assigned to base stations of the MEC network G. Each request is pre-assigned to G, following the similar steps in algorithm `Appro`. The differences lie in the steps after step 6 of `Appro`. Let r_j be the request that is considering for admission. If currently pre-assigned requests in a base station occupy resource more than $l \cdot C_l$, we migrate one task of a previously pre-assigned request in R^{pre} to its nearby base station. The steps of the algorithm are shown in algorithm 2, referred to as `Heu`.

C. Algorithm Analysis

We first show the solution feasibility of algorithm `Appro`, by showing **LP** is a feasible relaxation to the **ILP-RM** in Lemma 1.

Lemma 1: Let $LPOpt$ be the optimal solution to **LP**, and Opt be the optimal solution to the reward maximization problem with a set of non-preemptive AR requests (i.e., **ILP-RM**). The relaxation in **LP** is a valid relaxation of **ILP-RM**, and $LPOpt \geq Opt$.

Proof We first show that Opt is a feasible solution to **LP**. To this end, we need to show that solution Opt meets the constraints of (9), (10), and (11). Clearly, Opt meets constraints (9) and (11) because $x_{ji} = \sum_l y_{jil}$. To show that Opt also meets constraint (11), we consider some resource slot l of base station bs_i and a choice of base stations in the optimal solution Opt. Recall that the size and reward of implementing request r_j are not known in advance. The optimal solution Opt thus is a randomized policy. Consider a run of the optimal solution, we use x^*_{jil} and $s^*_{j,\rho}$ to denote the indicator variables that shows whether request r_j is assigned to the resource slot l of bs_i in solution Opt and r_j has a size of ρ, respectively. Let X_l be the random variable indicating the most recent request that is assigned before resource slot l. Although resources are slotted in each bs_i, it must be mentioned that request r_j occupies a

Algorithm 2 An efficient heuristic for the reward maximization problem, i.e., `Heu`

```
Input: An MEC network $G = (\mathcal{BS}, E)$, a set of requests with each
    request $r_j$ having its tasks being consolidated into a single base
    station, the data rate of each request is not known in advance
    until it is scheduled.
Output: An offloading decision for each request $r_j$.
1: Obtain a fraction solution $y$ by solving the LP;
2: Pre-assign request $r_j$ to resource slot $l$ of base station $bs_i$ with
   probability $\frac{y_{jil}}{4}$, and ignore $r_j$ with probability $1 - \frac{y_{jil}}{4}$;
3: $R^{Pre} \leftarrow \emptyset$; /* The set of requests that are preassigned to base
   stations of the MEC network $G$ */
4: for $l \leftarrow 1, \cdots, L$ do
5:     for each base station $bs_i \in \mathcal{BS}$ do
6:         Consider the request with the $l$th smallest data rate;
7:         if the requests assigned so far in $bs_i$ occupy at most $l \cdot C_l$
           amount of computing resource then
8:             Pre-assign $r_j$ to base station $bs_i$;
9:             $R^{Pre} \leftarrow R^{Pre} \cup \{r_j\}$;
10:        else
11:            For the requests that are already pre-assigned to base
               station $bs_i$, let $r_{j'}$ be the one with the maximum realized
               data rate;
12:            if the pre-assignment of a task of $r_{j'}$ reduces the
               accumulative resource consumption to below $l \cdot C_l$ then
13:                Pre-assign a task of $r_{j'}$ to the closest base station of
                   $bs_i$;
14:                Pre-assign $r_j$ to base station $bs_i$;
15:                $R^{Pre} \leftarrow R^{Pre} \cup \{r_j\}$;
16: Assign the tasks of requests according to the pre-assignment in
    $R^{pre}$;
```

consecutive amount of computing resource of bs_i, instead of scattered resource slots. We thus know that request X_l is the only request in Opt that may have its demanded computing resource exceeds that of resource slot l. This means that

$$\sum_{r_j \neq X_l} \sum_{\substack{l' \leq l \\ \rho \cdot C_{unit} \leq C(bs_i)}} x^*_{jil} \cdot s^*_{j,\rho} \cdot \rho \leq \frac{l \cdot C_l}{C_{unit}}. \tag{13}$$

If we include request X_l in Ineq. (13) and truncating the data rates by $\frac{l \cdot C_l}{C_{unit}}$, we then have

$$\sum_{r_j \in R} \sum_{l' \leq l} \sum_{\rho \in \mathcal{DR}} x^*_{jil} \cdot s^*_{j,\rho} \cdot \min\{\rho, \frac{l \cdot C_l}{C_{unit}}\} \leq \frac{2 \cdot l \cdot C_l}{C_{unit}}.$$

Recall that we consider algorithms that schedule a request before knowing its data rate. Clearly, we know x^*_{jil} and $s^*_{j,\rho}$ are independent variables. We thus re-write the LHS of Ineq. (14) as

$$\begin{aligned}
&\sum_{r_j \in R} \sum_{l' \leq l} \sum_{\rho \in \mathcal{DR}} Pr[x^*_{jil} = 1] \cdot Pr[s^*_{j,\rho} = 1] \cdot \min\{\rho, \frac{l \cdot C_l}{C_{unit}}\} \\
&= \sum_{r_j \in R} \sum_{l' \leq l} Pr[x^*_{jil} = 1] \sum_{\rho \in \mathcal{DR}} Pr[s^*_{j,\rho} = 1] \cdot \min\{\rho, \frac{l \cdot C_l}{C_{unit}}\} \\
&= \sum_{r_j \in R} \sum_{l' \leq l} x^*_{jil'} \cdot E(\min\{\rho, (l \cdot C_l)/C_{unit}\}).
\end{aligned} \tag{14}$$

where $Pr[Y]$ denotes the probability that event Y is true. Constraint (11) is met by the optimal solution Opt.

We then show that $LPOpt \geq Opt$. Assume that request r_j in the optimal solution is assigned at a starting resource slot l of base station bs_i, its obtained expected reward thus is

$$E(Opt_j \mid x^*_{jil} = 1) = ER_{jil}. \tag{15}$$

Considering all possible resource slots and base stations, it is not hard to obtain

$$E(Opt_j) = \sum_{l,i} x^*_{jil} ER_{jil}. \tag{16}$$

Therefore, we have $LPOpt \geq Opt$.

Lemma 2: The probability of failing to assign a request to its randomly selected starting resource slot of a base station is upper bounded by $\frac{1}{2}$.

Proof Let Y be a random variable that shows how much computing resource that is occupied by existing requests in the system. We need to show that

$$Pr[Y \geq l \cdot C_l] \leq 1/2. \tag{17}$$

Consider a request $r_{j'}$ that is already assigned to base station bs_i. Let $Y_{j'}$ be the amount of computing resource that is occupied by request $r_{j'}$. Denote by $\alpha_{\rho \cdot C_{unit} < l \cdot C_l}$ an indicator variable that shows whether $\rho \cdot C_{unit} < l \cdot C_l$. According to the steps in algorithm **Appro**, it is easy to obtain

$$Y_{j'} \leq \alpha_{\rho \cdot C_{unit} < l \cdot C_l} \cdot \sum_{\rho \in \mathcal{DR}} s_{j',\rho} \cdot \min\{\rho, (l \cdot C_l)/C_{unit}\},$$

where $s_{j',\rho}$ is an indicator variable that shows whether request $r_{j'}$ has a data rate of ρ.

Taking expectation on both sides of the above inequality, we get

$$\begin{aligned} E(Y_{j'}) &\leq E(\alpha_{\rho \cdot C_{unit} < l \cdot C_l}) \cdot E(\min\{\rho_{j'}, (l \cdot C_l)/C_{unit}\}), \\ &= \frac{1}{4} \cdot \sum_{l' \leq l} \sum_{bs_i \in \mathcal{BS}} y_{jil'} \cdot E(\min\{\rho_{j'}, (l \cdot C_l)/C_{unit}\}). \end{aligned}$$

Considering that $Y = \sum_{j'} Y_{j'}$, we have

$$E(Y) = \frac{1}{4} \sum_{j'} \sum_{l' \leq l} \sum_{bs_i \in \mathcal{BS}} y_{jil'} \cdot E(\min\{\rho_{j'}, \frac{l \cdot C_l}{C_{unit}}\}) \tag{18}$$

$$\leq (l \cdot C_l)/2, \text{due to constraint (10).} \tag{19}$$

Applying Markov's inequality, it is not hard to obtain

$$Pr[Y \geq l \cdot C_l] \leq 1/2. \tag{20}$$

This concludes the proof.

Theorem 1: Given an MEC network $G = (\mathcal{BS}, E)$, a set R of AR requests waiting to be admitted, there is an approximation algorithm, i.e., Appro, that produces a feasible solution with an approximation ratio of $\frac{1}{8}$ for the reward maximization problem with tasks of each request are consolidated into a single base station.

Proof We show the approximation ratio of the proposed algorithm Appro. Recall that algorithm Appro assigns request r_j to resource slot l of a base station with probability $\frac{y_{jil}}{4}$. However the request may be rejected if existing requests occupy more than the amount $l \cdot C_l$ of computing resource of the base station, in the following steps of the algorithm. As shown in Lemma 2, such an event probability is smaller than $1/2$. On the contrary, the probability of assigning it to base station bs_i is greater than $1/2$. Let V_j be the random variable the corresponds to the reward that algorithm Appro can obtain from admitting request r_j. We then have

$$E(V_j) \geq 1/8 \sum_l y_{jil} ER_{jil} = (1/8)LPOpt \geq (1/8)Opt,$$

where $LPOpt$ is the optimal solution to **LP**.

Theorem 2: Given an MEC network $G = (\mathcal{BS}, E)$, a set R of AR requests waiting for admissions, assume that the data rate of each request $r_j \in R$ is not known until it is assigned to a base station for execution, there is an efficient heuristic algorithm, i.e., Heu , that delivers a feasible solution for the reward maximization problem.

Proof We first show the solution feasibility of algorithm Heu. Recall that algorithm Appro is feasible to the reward maximization problem with a set of non-preempted requests. The solution obtained by algorithm Heu is based on the adjustment of algorithm Appro, by distributing the tasks of requests to their nearby base stations. The re-assignment of a task of request r_j makes sure its latency requirement and base station's capacity constraint are met. Algorithm Heu thus delivers a feasible solution.

We then analyze the time complexity of algorithm Heu that consists of two parts: (1) solving the **LP**, and (2) request pre-assignment. For part (1), it can be seem that **LP** has $O(|R| \cdot |\mathcal{BS}| \cdot L)$ decision variables. According to Cohen *et al.* [7], the running time of solving a linear program is $O((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}) I \log^k((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I))$, where n is the number of decision variables, I is the number of encoding bits of the variables, ω is the exponent of matrix multiplication and α is the dual exponent of matrix multiplication in Cohen's algorithm. The running time of solving **LP** in algorithm Heu by substituting n with $|R| \cdot |\mathcal{BS}| \cdot L$ can be deriven. For part (2), the checking of each pre-assignment obtained in part (1) is done maximally $O(|R| \cdot |\mathcal{BS}|)$ times. In summary, the running time of algorithm Heu is $O(|R| \cdot |\mathcal{BS}| + (n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}) I \log^k((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I))$, where $n = |R| \cdot |\mathcal{BS}| \cdot L$, where the current values of ω and α are 2.37 and 0.31, respectively.

V. Online Learning Algorithm for the Dynamic Reward Maximization Problem

AR requests in real MEC networks can be pre-emptive scheduled for later execution, for the sake of temporal fairness, considering limited computing resources of the MEC network, applying algorithms Appro and Heu at different time slots directly may increase the waiting time $b_j - a_j$ of request r_j with low rewards. We here propose efficient methods for such AR requests to avoid their scheduling starvation.

Algorithm 3 An online learning algorithm for the dynamic reward maximization problem, i.e., `DynamicRR`

```
Input: An MEC network $G = (\mathcal{BS}, E)$, a set of requests with each request $r_j$ having its tasks being consolidated into a single base station, the data rate of each request is not known in advance until it is scheduled.
Output: An offloading decision for requests in each time slot.
1: Divide the interval $Z$ into $\kappa$ intervals with fixed length $\epsilon = \frac{C^{th}_{max} - C^{th}_{max}}{\kappa - 1}$;
2: Let $Z'$ be the discretized set of arms;
3: for each time slot $t \leftarrow 1 \cdots T$ do
4:     Let $R'_t$ be the set of arrived requests;
5:     Try all active arms in $Z'$ in possibly multiple rounds;
6:     for each arm $a \in Z'$ do
7:         if there exists an arm $a'$ with $UCB_t(a) < LCB_t(a')$ then
8:             Deactivate arm $a$;
9:     Choose an active arm in $Z'$ that has the maximum reward, and use its value as the threshold $C^{th}_t$;
10:    Sort requests in $R'_t$ into increasing order of their expected data rates;
11:    Add requests in the sorted requests to $R_t$, until the average computing resource shared via RR is lower than $C^{th}$;
12:    Invoke algorithm Heu with the LP being replaced by LP-PT;
```

A. Online Learning via Lipschitz Bandits

To reduce the response time of each request, a naive approach is to schedule requests immediately after their arrivals. However, the system may not have enough computing resource to meet the total resource demand of the requests. We may allow equal resource sharing among the requests within each time slot. However, if there is a burst of requests to be scheduled in a particular time slot, the base stations can be overloaded if all the arrived requests equally share its resource. To avoid the over-congestion of base stations, we set a threshold on the minimum amount of computing resource to be assigned to each request at each time slot. Denote by C^{th} the threshold, which can be obtained and adjusted dynamically by the current latency of requests. Without loss of generality, assume that C^{th} is a value in the range of $[C^{th}_{min}, C^{th}_{max}]$.

Finding a suitable value of C^{th} is challenging since there basically are infinite numbers of values in the range of $Z = [C^{th}_{min}, C^{th}_{max}]$. The basic idea to identify a proper value of C^{th} is to discretize the continuous range into a finite set of candidate values for the threshold, and then adopt a multi-armed bandit method to dynamically select a value for C^{th} that could achieve the best reward, which is described in the following.

Given the range Z of C^{th}, consider each value in Z as a potential arm. Z thus is the set of potential arms. Let $ER(a)$ be the expected reward of selecting an arm a in Z. We assume that the expected reward satisfies the following Lipschitz condition:

$$|ER(a) - ER(b)| \leq \eta |a - b|, \text{for any arms } a, b \in Z, \quad (21)$$

where η is a given constant.

We perform discretization of the value interval Z by dividing it into κ subintervals with fixed length $\epsilon = \frac{C^{th}_{max} - C^{th}_{max}}{\kappa - 1}$, so that the obtained set Z' consists of all integers multiplying by ϵ. Clearly, $Z' \subset Z$. After the discretization of Z, a finite number of arms in set Z' is obtained. Let $ER^*(Z')$ be the best arm in Z' that achieves the maximum reward of scheduling the requests. The proposed MAB algorithm aims to approach the expected reward $ER^*(Z')$. To this end, we adopt a successive elimination algorithm. That is, all the arms in Z' are set to "active" initially. In each time slot t, we assume that the upper and lower bounds on the reward of each arm a exist. Denote by $UCB_t(a) = ER_t(a) + r_t(a)$ and $LCB_t(a) = ER_t(a) - r_t(a)$, the upper and lower bounds on the reward of arm a at time slot t, where $r_t(a)$ can be considered as the confidence radiation. We then examine all active arms, and deactivate all arms a if there is an arm $a' \in Z'$ with $UCB_t(a) < LCB_t(a')$. This procedure continues until there is only arm left in Z', which is selected as the threshold C^{th}_t at time slot t.

Given a threshold C^{th}_t, we now conduct the similar relaxation as we did for **LP** at each time slot t.

$$\textbf{LP-PT:} \quad \max \sum_{jil} y_{jil} \cdot ER_{jil},$$

subject to constraints (11), (12), and

$$\sum_{il} y_{jil} \leq 1, \forall r_j \in R_t \quad (22)$$

$$\sum_{jil':l' \leq l} y_{jil'} E(\min\{C(bs_i)/|R_t|, \rho_j, l \cdot C_l/C_{unit}\}) \leq 2 \cdot l \cdot C_l/C_{unit}, \forall l \quad (23)$$

where $|R_t|$ is the number of requests to be scheduled for execution at time slot t. We determine R_t by the expected data rate of the arrived request at time slot t. Specifically, we sort the arrived requests in increasing order of their expected data rate. We then add the requests into R_t one by one. Note that a larger set of R_t usually means that less resource will be allocated to each request. We thus keep picking requests in the sorted list and adding requests to R_t, until the addition of a request makes each request being assigned with an amount less than C^{th} amount of computing resource.

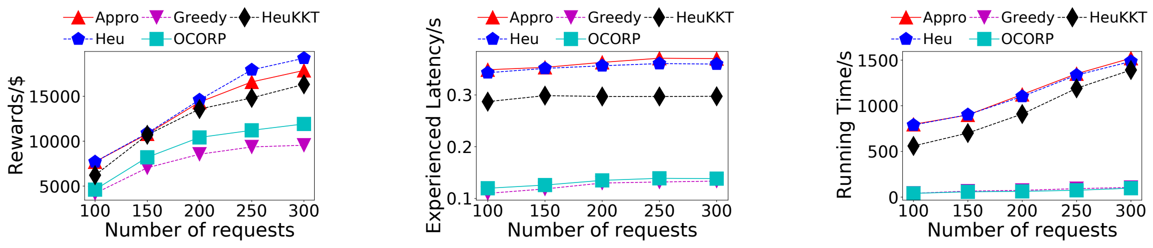
The request assignment at each time slot t then follows the similar method as that of algorithm `Heu`. The detailed steps of the proposed online algorithm is given in **Algorithm** 3, which is referred to as `DynamicRR`.

Theorem 3: The regret of **Algorithm** 3 is $O(\sqrt{\kappa T \log T} + T \cdot \eta \cdot \epsilon)$, where κ is the number of intervals that Z is discretized, $\epsilon = \frac{C^{th}_{max} - C^{th}_{max}}{\kappa - 1}$, T is the entire time horizon, and η is the given constant of the Lipschitz condition in Eq. (21).

Proof We first define the expected reward of algorithm 3. In algorithm 3, we discretize the continuous set of arms Z to Z' with κ arms. It is clear that adding more points in Z' makes a better approximation of Z; it however may increase the regret as well. To characterize the award loss(or the regret) by adopting Z' instead of Z, the *discretization error* is adopted, that is

$$DE(Z') = ER^*(Z) - ER^*(Z'). \quad (24)$$

If a^* is the best arm in Z, and b is the closest arm in Z' to



(a) Total rewards. (b) Average latency. (c) Running time.

Fig. 3. The performance of algorithms `Appro`, `Heu`, `Greedy`, `OCORP` and `HeuKKT`.

a^* in Z, we have $|a^* - b| \leq \epsilon$. By the Lipschitz condition in Eq. (21), we then have

$$DE(Z') \leq \eta\epsilon. \tag{25}$$

Denote by $W(\texttt{DynamicRR})$ the total reward obtained by algorithm `DynamicRR`. The expected regret of algorithm `DynamicRR` in time horizon T is defined as

$$\begin{aligned} E(R(T)) &= T \cdot ER^*(Z) - W(\texttt{DynamicRR}) \\ &= (T \cdot ER^*(Z') - W(\texttt{DynamicRR})) + \\ &\quad T \cdot (ER^*(Z) - ER^*(Z')) \\ &= R_S(T) + T \cdot DE(Z'), \end{aligned} \tag{26}$$

where $R_S(T)$ is the regret relative to $ER^*(Z')$ obtained due to the successive elimination arm selection procedure.

Following the work in [25], the regret bound of a successive elimination algorithm is $O(\sqrt{\kappa T \log T})$, which means that

$$R_S(T) = O(\sqrt{\kappa T \log T}). \tag{27}$$

By (27) and (25), we have the regret bound of algorithm `DynamicRR` as $E(R(T)) \leq O(\sqrt{\kappa T \log T} + T \cdot \eta \cdot \epsilon)$.

VI. Experiments

A. Parameter Settings

We consider an MEC network with 20 base stations, whose topology is generated using GT-ITM [13]. Each station has a computing capacity that is randomly drawn from the range of [3,000 to 3,600] MHz and each resource-slot owns 1,000 MHz amount of computing resource [28]. The data rate of each is randomly drawn from 10 to 15 Mbps, and each request has 3 to 5 tasks. The maximum number of requests is 150. For AR requests, we adopt the real dataset in [5], which is collected in real environments by adopting OpenCV6 for tracking and YOLO for recognizing objects. Specifically, each AR request processes a series of JPEG images. The size of each image is set to 64Kb, and the uploading rate is 90-120 frames per second [5]. Each AR request includes four tasks: the render object (100Kb), track objects (64Kb), update world model (64Kb) and the recognize objects (64Kb). As such, we set the data rate of a request is drawn from the range of [30, 50] Mega Bytes (MB) per second [5]. Assuming that processing every each MB data will consume 20 MHz computing resource. The maximum response delay is less than 200 milliseconds [18] and the length of each time slot is 0.05 seconds. The reward serving a unit amount of data rate is in the range of [12, 15] dollars [24]. Unless otherwise specified, these parameters will be applied as default settings.

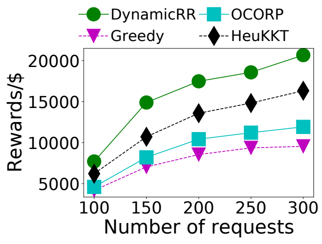
We compare the proposed algorithms against the following benchmark algorithms.

- We first compare the proposed algorithms with an algorithm that adopts online convex optimization in [20], which is referred to as `OCORP`. Specifically, in each time slot, algorithm `OCORP` sorts the unfinished job according to arriving time and remaining to-be-processed data, then assigns tasks to edge servers based on a best-fit algorithm.
- We then compare the algorithms with a heuristic algorithm in [32], referred to as `Greedy`. The algorithm sorts tasks in a decrease order according to their execution times, and assign the task to the optimal edge server one-by-one.
- We also compare the proposed algorithms with another heuristic in [21], which is referred to as `HeuKKT`. The algorithm first removes the constraints of resource capacities to find the workload offloaded to remote cloud. It then finds the optimal scheduling solutions in edge servers fitting Karush-Kuhn-Tucker (KKT) conditions with resource constraints.

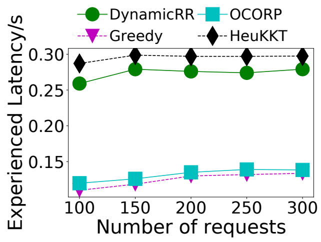
In our experiments, these benchmarks are implemented as offline and online versions to compare with the proposed offline algorithms `Appro`, `Heu`, and online algorithm `DynamicRR`.

B. Performance Evaluation

We first study the performance of algorithms `Appro`, `Heu`, `OCORP`, `Greedy`, and `HeuKKT` in terms of the total rewards, average latencies of a request, and running times, by varying the number of requests from 100 to 300. From Fig. 3 (a) and (c), we can see that algorithm `Appro` achieves 50%, 80%, 9% higher rewards than those of algorithms `OCORP`, `Greedy`, and `HeuKKT`, although it has a slightly higher running time than the rest algorithms. Also, algorithm `Heu` has 61%, 101%, and 17% higher rewards than those of algorithms `OCORP`, `Greedy`, and `HeuKKT` when the number of requests is 300. The reason is that algorithms `Appro` and `Heu` carefully deal with the uncertainty of data rates of requests, by considering a resource-slot based allocation. Instead, algorithms `OCORP`, `Greedy`, and `HeuKKT` adopt coarse-grained methods according to the data rates of requests, execution times, and the resource capacity constraints of edge servers. We can also see that the average latencies

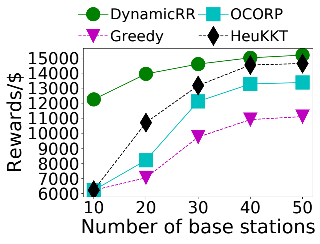


(a) Total rewards.

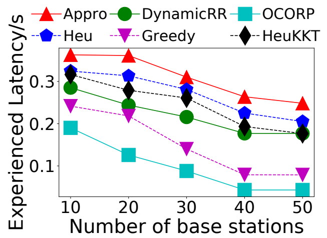


(b) Average latency.

Fig. 4. The performance of algorithms `DynamicRR`, `Greedy`, `OCORP` and `HeuKKT`.

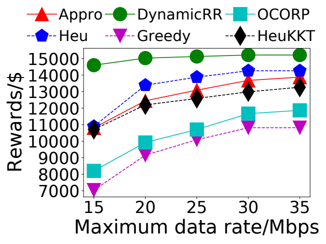


(a) Total rewards.

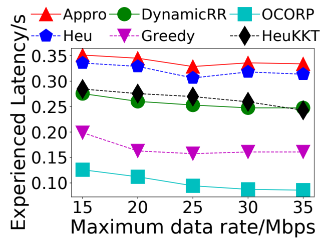


(b) Average latency.

Fig. 5. The performance of algorithms `Appro`, `Heu`, `DynamicRR`, `Greedy`, `OCORP`, and `HeuKKT` by varying $|R|$ from 80 to 200.



(a) Total rewards.



(b) Average latency.

Fig. 6. The performance of algorithms `DynamicRR`, `Greedy`, `OCORP` and `HeuKKT` by varying max $|DR|$.

of algorithms `OCORP` and `Greedy` are lower than that of the rest algorithms, because they trade-off the reward for latency. Although algorithms `Appro`, `Heu`, and `HeuKKT` have higher latencies, they have much higher rewards than the rest of the comparison algorithms.

We then study the performance of algorithms `DynamicRR`, `OCORP`, `Greedy`, and `HeuKKT` in terms of the total rewards and average latencies of a request, by varying the number of requests from 100 to 300. It can be seen from Fig. 4 (a), algorithm `DynamicRR` has a higher reward and lower average latency than those of algorithm `HeuKKT`. The rationale behind is that algorithm `DynamicRR` adopts a MAB-based threshold adjusting method to avoid the starvation of AR requests with low rewards. Also, the rewards of the algorithms first increase and then keep stable with the growth of the number of requests. This is because as the number of requests increases, the network resource becomes saturated and no more requests can be admitted. As shown in Fig. 4 (b), algorithm `DynamicRR` has a higher latency than that of algorithms `OCORP` and `Greedy`, because algorithms `OCORP` and `Greedy` greedily select locations that achieve the lowest latencies. Also, the experienced latencies of algorithm `OCORP` and `Greedy` increase with the growth of the number of requests, while other algorithms keep stable. This is because they utilize a local strategy instead of considering the global optimal solution.

C. Impact of Parameters

We then investigate the impact of the number of base stations $|\mathcal{BS}|$ on the performance of algorithms `Appro`, `Heu`, `DynamicRR`, `OCORP`, `Greedy`, and `HeuKKT`, by varying $|BS|$ from 10 to 50. The results of the total reward and the average latency of a request are shown in Fig. 5, from which we can see that the total reward is increasing with the growth of $|BS|$. The reason is that with the growth on the number of base stations, requests can all be scheduled to the base stations with higher rewards. Instead, as shown in Fig. 5 (b), the average latency decreases with the growth of base station numbers. The reason is that the requests have a higher probability of being scheduled into base stations with low processing latencies.

We finally investigate the maximum data rate of a request on the performance of algorithms `Appro`, `Heu`, `DynamicRR`, `OCORP`, `Greedy`, and `HeuKKT`, by varying the maximum data rate of a request from 15 to 35 Mbps. From Fig. 6, we can see that the reward grows with the increase of the maximum data rate. In addition, the latency increases with the growth of the maximum data rate. The rationale behind is that a larger data rate usually implies more processing time needed.

VII. Conclusion

In this paper, we studied the reward maximization problem for AR applications in an MEC network. For the problem with a set of non-preemptive scheduling of AR requests, we devised an exact solution and an approximation algorithm with an approximation ratio. We also developed an online learning algorithm for the dynamic reward maximization problem if AR requests can be allowed to be preemptively scheduled, by leveraging the techniques of round-robin and multi-armed bandits. We finally evaluated the performance of the proposed algorithms by simulations. Experimental results show that the reward obtained by the proposed algorithms outperform that of existing studies by at least 17% in terms of the total award.

Acknowledgements

The work by Zichuan Xu and Qiufen Xia is partially supported by the National Natural Science Foundation of China (Grant No. 61802048 and 61802047), the fundamental research funds for the central universities in China (Grant No. DUT19RC(4)035), and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work by Weifa Liang is supported by the Australian Research Council Discovery Project (Grant No. DP200101985). The work done by Pan Zhou is supported in part by NSFC (Grant No. 61972448).

References

[1] A. Badshah, A. Ghani, S. Shamshirband, and A. T. Chronopoulos. Optimising infrastructure as a service provider revenue through customer satisfaction and efficient resource provisioning in cloud computing. *IET Communications*, vol. 13, no. 18, pp. 2913–2922, IET, 2019.

[2] Z. Bhatti, M. Bibi, and N. Shabbir. Augmented reality based multimedia learning for dyslexic children. *Proc. of iCoMET*, IEEE, 2020.

[3] S. Bi, and Y. J. Zhang. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, IEEE, 2018.

[4] S. Bohez, J. D. Turck, T. Verbelen, P. Simoens, and B. Dhoedt. Mobile, collaborative augmented reality using cloudlets. *Proc. of MOBILWARE*, IEEE, 2013.

[5] T. Braud, P. Zhou, J. Kangasharju, and P. Hui. Multipath computation offloading for mobile augmented reality. *Proc. of PerCom*, IEEE, 2020.

[6] W. Chang, Y. Xiao, W. Lou, and G. Shou. Offloading decision in edge computing for continuous applications under uncertainty. *IEEE Transactions on Wireless Communications*, Vol. 19, No. 9, pp. 6196–6209, 2020.

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

[8] P. Cong, G. Xu, T. Wei, and K. Li. A survey of profit optimization techniques for cloud providers. *ACM Computing Serveys*, vol. 53, no. 2, article no. 26, pp. 1–35, ACM, 2020.

[9] J. Du, E. Gelenbe, C. Jiang, Z. Han, and Y. Ren. Auction-based data transaction in mobile networks: Data allocation design and performance analysis. *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1040–1055, IEEE, 2019.

[10] N. Eshraghi, and B. Liang. Joint offloading decision and resource allocation with uncertain task computing requirement. *Proc. of INFOCOM*, IEEE, 2019.

[11] V. Farhadi, F. Mehmeti, T. He, T. La Porta, H. Khamfroush, S. Wang, and K. S. Chan. Service placement and request scheduling for data-intensive applications in edge clouds. *Proc. of INFOCOM*, IEEE, 2019.

[12] M. Gattullo, A. Evangelista, A. E. Uva, M. Fiorentino, and J. Gabbard. What, how, and why are visual assets used in industrial augmented reality? A systematic review and classification in maintenance, assembly, and training (from 1997 to 2019). *IEEE Transactions on Visualization and Computer Graphics*, DOI: 10.1109/TVCG.2020.3014614, 2020.

[13] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[14] M. Hadji, and D. Zeghlache. Mathematical programming approach for revenue maximization in cloud federations. *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 99–111, IEEE, 2015.

[15] A. Jakl, A. M. Lienhart, C. Baumann, A. Jalaeefar, A. Schlager, L. Schöffer, and F. Bruckner. Enlightening patients with augmented reality. *Proc. of VR*, IEEE, 2020.

[16] H. J. Jeong, H. J. Lee, C. H. Shin, and S. M. Moon. IONN: incremental offloading of neural network computations from mobile devices to edge servers. *Proc. of SoCC*, ACM, 2018.

[17] S. Jošilo, and G. Dán. Wireless and computing resource allocation for selfish computation offloading in edge computing. *Proc. of ICCC*, IEEE, 2019.

[18] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. *Proc. of MOBICOM*, ACM, 2019.

[19] W. Liu, J. Ren, G. Huang, Y. He, and G. Yu. Data offloading and sharing for latency minimization in augmented reality based on mobile-edge computing. *Proc. of VTC*, IEEE, 2018.

[20] Y. Liu, H. Xu, and W. C. Lau. Online job scheduling with resource packing on a cluster of heterogeneous servers. *Proc. of ICCC*, IEEE, 2019.

[21] X. Ma, A. Zhou, S. Zhang, and S. Wang. Cooperative service caching and workload scheduling in mobile edge computing. *Proc. of ICCC*, IEEE, 2020.

[22] X. Qiao, P. Ren, G. Nan, L. Liu, S. Dustdar, and J. Chen. Mobile web augmented reality in 5G and beyond: Challenges, opportunities, and future directions. *China Communications*, vol. 16, no. 9, pp. 141–154, IEEE, 2019.

[23] J. Ren, G. Yu, Y. Cai, and Y. He. Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, IEEE. 2018.

[24] X. Shao, G. Hasegawa, N. Kamiyama, Z. Liu, H. Masui, and Y. Ji. Joint optimization of computing resources and data allocation for mobile edge computing (MEC): An online approach. *Proc. of ICCCN*, IEEE, 2019.

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

[26] T. X. Tran, and D. Pompili. Adaptive bitrate video caching and processing in mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, IEEE, 2018.

[27] H. Wang, and J. Xie. User preference based energy-aware mobile AR system with edge computing. *Proc. of ICCC*, IEEE, 2020.

[28] Z. Xu, L. Zhou, S. C. K. Chau, W. Liang, Q. Xia, and P. Zhou. Collaborate or separate? Distributed service caching in mobile edge clouds. *Proc. of INFOCOM*, IEEE, 2020.

[29] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, Efficient algorithms for capacitated cloudlet placements, *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, IEEE, 2016.

[30] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu. Energy-aware inference offloading for DNN-driven applications in mobile edge clouds. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 4, pp. 799-814, IEEE, 2021.

[31] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, Vol.18, No. 11, pp. 2672-2685, IEEE, 2019.

[32] T. Yang, R. Chai, and L. Zhang. Latency optimization-based joint task offloading and scheduling for multi-user MEC system. *Proc. of WOCC*, IEEE, 2020.

[33] A. Younis, T. X. Tran, and D. Pompili. Energy-latency-aware task offloading and approximate computing at the mobile edge. *Proc. of MASS*, IEEE, 2019.

[34] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang. Offloading dependent tasks in mobile edge computing with service caching. *Proc. of INFOCOM*, IEEE, 2020.

[35] Verizon. *https://www.thefastmode.com/technology-solutions/15725-verizon-develops-new-5g-edge-technology-for-vr-mr-and-ar.*