Learning-Driven Algorithms for Responsive AR Offloading With Non-Deterministic Rewards in Metaverse-Enabled MEC

Zichuan Xu, *Member, IEEE*, Zhao Yuan, Weifa Liang, *Senior Member, IEEE*, Dongqi Liu, Wenzheng Xu, *Member, IEEE*, Haipeng Dai, *Senior Member, IEEE, Member, ACM*, Qiufen Xia, *Member, IEEE*, and Pan Zhou, *Member, IEEE*

***Abstract*— In the coming era of Metaverse, Augmented Reality (AR) has become a key enabler of diverse applications including healthcare, education, smart cities, and entertainments. To provide users with interactive and immersive experience, most AR applications require extremely high responsiveness and ultra-low processing latency. Mobile edge computing (MEC) has demonstrated great potentials in meeting such stringent latency requirements and resource demands of AR applications, by implementing AR requests in edge servers within the proximity of users. In this paper, we investigate the reward maximization problem for AR applications with uncertain resource demands in an MEC network, such that the accumulative reward of services provided for AR applications is maximized, while ensuring that the responsiveness of AR applications is enhanced, subject to network resource capacity. To this end, we formulate an exact solution when the problem size is small, otherwise we devise an efficient approximation algorithm with a provable approximation ratio for the problem. We also develop an online learning algorithm with a bounded regret for the dynamic reward maximization problem without the knowledge of future arrivals of AR requests, by adopting the Multi-Armed Bandits (MAB) technique. Considering maximizing the reward may defer the implementations of some urgent yet low-award requests, we propose a fairness-aware online learning algorithm for the dynamic reward maximization problem, through a data rate prediction mechanism that adopts a multi-task and multi-timescale Long Short-Term Memory (MT^2-LSTM) method. Finally, we evaluate the performance of the proposed algorithms for AR applications by building a real test bed. Experimental results show that the proposed algorithms outperform existing studies by improving the award by 13%.**

***Index Terms*— Mobile edge computing, augmented reality, reward maximization, online learning, multi-armed bandits.**

Manuscript received 26 January 2023; revised 20 August 2023; accepted 28 September 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Pedarsani. Date of publication 18 December 2023; date of current version 18 April 2024. The work of Zichuan Xu and Qiufen Xia was supported by the National Natural Science Foundation of China (NSFC) under Grant 62172068, Grant 62172071, and Grant T2350710232. The work of Weifa Liang was supported by the City University of Hong Kong under Project 9380137/CS. The work of Wenzheng Xu was supported by the NSFC under Grant 62272328. The work of Pan Zhou was supported by the NSFC under Grant 61972448. *(Corresponding author: Wenzheng Xu.)*

Zichuan Xu, Zhao Yuan, and Dongqi Liu are with the School of Software, Dalian University of Technology, Dalian 116620, China, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China (e-mail: z.xu@dlut.edu.cn; yuanzhao@mail.dlut.edu.cn; ldq0913@mail.dlut.edu.cn).

Weifa Liang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: weifa.liang@cityu.edu.hk).

Wenzheng Xu is with the College of Computer Science, Sichuan University, Chengdu, Sichuan 610065, China (e-mail: wenzheng.xu@scu.edu.cn).

Haipeng Dai is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: haipengdai@nju.edu.cn).

Qiufen Xia is with the International School of Information Science and Engineering, Dalian University of Technology, Dalian 116620, China, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China (e-mail: qiufenxia@dlut.edu.cn).

Pan Zhou is with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: panzhou@hust.edu.cn).

Digital Object Identifier 10.1109/TNET.2023.3323514

I. INTRODUCTION

AUGMENTED Reality (AR) is emerging as a key technology in Metaverse, by connecting the virtual cyber world and the real physical world to provide users with an immersive experience [29]. In particular, it inserts virtual content into a stream of views of the real world captured from the physical environment, by various sensors and cameras [17], [22]. An AR application captures the physical environment and sends its video streams to a mobile edge computing (MEC) network at a certain data rate for inference. 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 [31]. This consequently downgrades the immersive experience of Metaverse.

With the fast development of beyond 5G and 6G communications, MEC is envisioned as the key enabling technology for Metaverse [29], [48]. In an MEC network, the content and computing resources are close to end users, providing extreme low-latency services [23], [31]. Therefore, various Metaverse service providers are seeking opportunities of promoting their revenues by deploying AR services with strong real-time guarantees in the network edge. For example, Verizon 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 [50]. In particular, Metaverse service providers can place their AR services to locations in an MEC network and can offload user tasks to the placed AR services for implementations, which is referred to as *task offloading*.

Task offloading of AR applications in an MEC network poses several crucial challenges.

First, the objective of Metaverse service providers by offloading AR requests to the MEC network is to maximize their revenues. Quantifying the rewards is important and challenging for Metaverse service providers to guide their marketing strategies. Each Metaverse service provider usually cannot determine the reward obtained by offloading an AR request in the MEC network solely based on the rates of data streams of the AR application. The reason is that the amount of reward received depends on multiple factors such as responsiveness, resource usage costs, processing delays, and pricing methods adopted by the Metaverse service provider [12]. Also, the actual rate of the data stream of an AR application is normally not known in advance before the AR request admission. Therefore, the deterministic relationship between the reward received and the data rates, such as a proportional model, is not realistic. How to offload requests of AR applications based on non-deterministic rewards is challenging.

Second, 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 aforementioned non-deterministic rewards, how to jointly improve the responsiveness of AR applications in an MEC network while meeting their delay requirements such that the accumulative reward is maximized is challenging.

To address the challenges, we investigate the accumulative reward maximization problem for AR applications with non-deterministic rewards in an MEC network. We aim to enhance the responsiveness of AR applications via offloading AR requests to the MEC network.

Most extensive studies on task offloading and service placement in MEC are based on conventional task models [1], [14], [20], [28], [35], [40], [44], [46], [51], [57], [59], [61], where proportional reward function models are adopted [9], [16], [39], [54]. Besides, most of these task offloading ignored the responsiveness needed in AR applications [8], [34], [48], [52]. To the best of our knowledge, we are the first to consider task offloading of AR requests in MEC with uncertain data rates, rewards, and demand-independent rewards.

The main contributions of this paper are as follows.

- We propose exact and approximation algorithms for the reward maximization problem for a set of non-preemptive AR requests with non-deterministic rewards, assuming that the tasks of each AR request can be consolidated into a single base station for processing.
- We develop an efficient heuristic for the reward maximization problem for a set of non-preemptive AR requests with non-deterministic rewards, if the tasks of each AR request can be distributed into multiple base stations.
- We devise an online learning algorithm with a bounded regret for the dynamic reward maximization problem for a sequence of preemptive AR requests with non-deterministic rewards, to jointly optimize the waiting time before scheduling and the processing delay for implementing each AR request, assuming that AR requests arrive into the system dynamically and admitted requests can be preempted for later implementations.
- We propose a fairness-aware online algorithm for the dynamic reward maximization problem with non-deterministic rewards if fairness among AR requests have to be considered, for which we propose a novel multi-task and multi-timescale LSTM (i.e., MT^2-LSTM) method.
- We build a real test bed to evaluate the performance of the proposed algorithms. Experimental results show that the proposed algorithms outperform their counterparts by improving the accumulative award by at least 13%.

This paper is organized as follows. Section II surveys related studies. Section III introduces the system model and defines the problems. Section IV proposes the exact and approximation algorithms for the reward maximization problem for a set of non-preemptive AR requests. Section V devises an online learning algorithm for the dynamic reward maximization problem. Section VI proposes a fair online learning algorithm for the dynamic reward maximization problem. Section VII evaluates the performance of the proposed algorithms in a real testbed, and Section VIII concludes the paper.

II. Related Work

Task offloading and service placement in MEC networks have been extensively explored to improve the latency of mobile services [14], [20], [28], [30], [35], [38], [40], [44], [60]. Most of the studies focused on conventional requests, assuming that each request is indivisible, which cannot be applied to AR requests with a sequence of tasks. For example, Li et al. [28] proposed task offloading policies to avoid resource over-distribution through deep reinforcement learning-based resource reservation and server cooperation, with the aim of maximizing the long-term offloading benefits on latency and energy consumption. Shang et al. [44] designed an online algorithm for service placement to enhance the quality of experience, by considering the mobility of end users and volatility of network conditions.

There are several studies of offloading requests with each having a sequence of dependent tasks [1], [25], [40], [42], [46], [51], [57], [60], [61]. Most of them however assumed that the resource demands of requests are given in advance. Considering that AR applications dispatch their tasks in the form of data streams, the data rates and total resource demand are unknown. The approaches of these existing studies thus are unlikely to be applicable to the task offloading of AR applications. For example, Xiao et al. [57] considered the problem of parallel task offloading and content caching, with the aim to minimize the task delay and energy consumption. Zhao et al. [61] investigated the problem of offloading dependent tasks with service placement to minimize the makespan of offloaded tasks. Wang et al. [51] investigated a reinforcement learning schema to maximize the quality of service for dependent tasks in an MEC network. Abbas et al. [1] studied a task completion maximization problem in a device-to-device enabled MEC network. Song et al. [46] studied an offloading scheme to maximize the reward of servers with a power budget.

There are also several investigations focused on the problem of reward maximization in MEC networks or cloud computing environments [9], [13], [16], [39], [54]. Nevertheless, most reward functions adopted in these studies are proportional to the demands of services or given in advance. For example, Ma et al. [39] investigated a revenue maximization problem by proposing an online algorithm with the constraint of system performance. Chen [9] derived a game theory based algorithm for the problem of pricing and offloading in an MEC network. Wang et al. [54] studied the problem of decentralized task offloading for a dynamic MEC network with the unknown system-side information, by developing a decentralized online learning algorithm with a sub-linear regret on rewards. Li et al. [36] formulated a network utility maximization problem through considering the uncertainty of request demands in the dense MEC network, by designing an online algorithm based on a two-timescale Lyapunov optimization.

On the other hand, although the problem of task offloading of AR applications has gained a few attentions [6], [8], [11], [18], [24], [34], [48], [52] recently, none of the mentioned works investigated the optimization of responsiveness of AR applications, not to mention uncertain user demands. Bohez et al. [5] studied task offloading for AR applications in MEC networks for the first time. They developed a platform that is capable of autonomously deploying software components to minimize the average CPU load while guaranteeing smooth collaborations. Chen et al. [8] formulated a joint optimization problem of task offloading and resource allocation for AR applications so that the energy consumption of each user is minimized. Zhang et al. [34] dealt with the placement of rendering tasks and encoding tasks for multi-user AR requests, to maximize the quality of experience while minimizing the cost of resources. Si et al. [48] investigated the problem of video resolution and resource allocation for mobile AR applications. Wang et al. [52] proposed an optimization algorithm to enable energy-efficient AR configuration adaptations and radio resource allocations in edge servers.

In addtion, the fairness issue on task offloading and resource allocation in MEC networks has also been dealt in the past [2], [3], [4], [21], [26], [47]. However, most of the studies only considered the fairness of generic requests in cloud computing and edge computing, and ignored the temporal fairness on the scheduling of AR requests with a sequence of tasks and uncertain data rates. For example, Baruah et al. [2] considered the temporal fairness problem in periodic real-time scheduling. Bei et al. [3] proposed a new metric, called fair-ratio, to satisfy the same set of properties with dominant resource fairness but with better efficiency guarantees. Bi et al. [4] proposed a fair task offloading algorithm to improve quality of service of vehicles. Shahsavari et al. [47] studied opportunistic user scheduling in a wireless network under short-term and long-term temporal fairness constraints.

This work is an extension of a conference paper [55], by considering fairness and conducting experiments in a test bed.

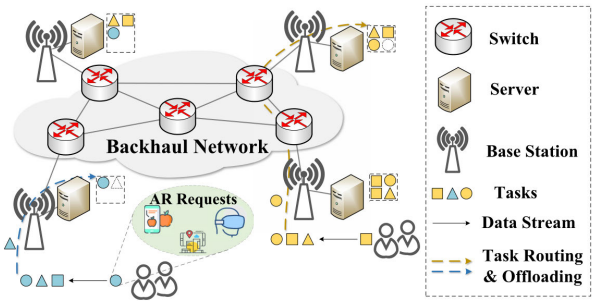


Fig. 1. An example of an MEC network.

III. System Model and Problem Definitions

A. System Model

We consider an MEC network $G = (\mathcal{BS}, E)$, where $\mathcal{BS}$ is a set of base stations, and E is the set of links/paths interconnecting the base stations in the backhaul of the MEC network. Each base station $i \in \mathcal{BS}$ is equipped with a certain amount of computing resource, such as a neural network accelerator and FPGA, to process AR requests. Let $C(i)$ be the computing capacity on each base station i. A link/path connects two base stations in the backhaul of the MEC network G. Let e be such a link/path in E.

An AR application usually runs in a user device, and sends inference requests to base stations of the MEC network for continuously processing of user data. We consider that each AR application sends its inference requests via its closest base station, to reduce the transmission latency of data. However, a base station may not be able to implement all inference requests, due to its limited computing capacity. We thus consider that the inference requests of each AR application can be distributed to multiple base stations in $\mathcal{BS}$ via the backhaul paths of G. Fig. 1 shows an example of the system model.

B. AR Requests and Services

We consider an AR application that analyzes the surroundings of a user and combine temporal information and spatial images captured by the application. Each AR application has a processing pipeline that renders virtual objects to a video stream. Following existing studies [6], [8], [34], [48], [52], each AR processing pipeline can include the tasks of pose estimation, mapping, world model creation, and rendering, where rendering processing is the most computing-intensive task. An AR processing pipeline thus is divided into a sequence of tasks. Each task of the sequence takes the output matrix of its predecessor as its input matrix, and produces its output matrix as its successor task. Let r_j be a request of an AR application and $\{(j,1), \cdots, (j,k), \cdots (j,K_j)\}$ be the sequence of tasks of r_j, where K_j is a positive integer and $1 \leq k \leq K_j$.

The video stream of an AR application is streamed to its task sequence for processing by the tasks of an AR processing pipeline. Let ρ_j be the data rate of the video stream of AR request r_j. In fact, the data rate of an AR request is determined by many factors, including the network condition, physical environment, data sampling methods of user devices, and so on. Such a data rate cannot be obtained in advance. However,

since the type of an AR application can be known in advance, i.e., gaming or shopping services, we assume that the data rates of requests of an AR application follow a probability distribution. 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 needed to process a unit data rate.

AR tasks need to be executed by AR services that are placed in the MEC network. Let S_j be the service required by request r_j. An *instance* of 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. In MEC networks, services are placed to base stations close to users, thereby reducing the delay of tasks. Considering that each instance of S_j usually is implemented in a Virtual Machine (VM) or a container, we assume that the computing resource allocated to each VM is given as a priori; for example, each VM or container may be assigned a CPU core and a neural network accelerator. We further assume that an instance of S_j is created for the implementation of each request r_j and then destroyed after its departure.

C. Non-Deterministic Rewards of Processing AR Requests

A Metaverse service provider receives a reward for the implementation of an AR request in an MEC network. The reward of implementing request r_j is related to its data rate ρ_j. The rationale behind is that the data rate determines both the resource consumption and the payment that the network service provider receives. However, the data rate of a request usually is not known in advance, and this leads to a non-deterministic reward. That is why different Metaverse service providers may have different pricing and cost models, so are resource usage costs and electricity costs. We thus assume that there is a probability distribution over its data rate and the reward for each AR request r_j. That is, the probability of its data rate being ρ for request r_j is $\pi_{j,\rho}$. Once the data rate ρ is obtained, the reward is $RD_{j,\rho}$. For simplicity, we use $(\pi_{j,\rho}, RD_{j,\rho})$ to denote the probability of the data rate ρ of request r_j, and $RD_{j,\rho}$ denotes the reward of implementing r_j, respectively.

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. The system then decides when to schedule request r_j. Denoted by a_j and b_j the arrival time slot of request r_j and the time slot that the first frame of r_j's video stream is processed, respectively. The delay of waiting for scheduling request r_j thus is $b_j - a_j$.

Once request r_j is scheduled, its video stream is transmitted to an instance of its service S_j for processing. Such data transmission incurs the transmission delay. Video processing by the service leads to the processing delay. The service of each request needs to process 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 to each video frame. Denote by ρ_{unit} the minimum size of video frames 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 are different. Let d^{pro}_{jki} be the delay of processing ρ_{unit} amount of data by task (j,k) of r_j in base station i. Similarly, d^{trans}_{je} represents the delay of transmitting the amount ρ_{unit} 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 delays. Each AR request has a requirement on its experienced latency. Let $\hat{D}_j$ be the latency requirement of request r_j. The experienced latency of request r_j must be no greater than its specified requirement $\hat{D}_j$, i.e., $D_j \leq \hat{D}_j$.

E. Problem Definitions

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

Problem 1: Assuming that AR requests are non-preemptive, *the reward maximization problem with non-deterministic rewards* aims to maximize the expected accumulative reward of implementing the requests in R, by placing the service instances to base stations and assigning the tasks of each request to its placed service instances while meeting the latency requirement of each admitted request, subject to the computing capacity on each base station.

Problem 2: Assuming that running tasks can be preempted and resumed their implementations later, *the dynamic reward maximization problem with non-deterministic rewards* for a given monitoring period T is to maximize the expected accumulative reward of admitted requests for the time period T while meeting the latency requirements of admitted requests, by placing the service instances to base stations and scheduling the tasks of each request to its placed service instance over the time period T, subject to the computing capacity on each base station.

IV. Approximation Algorithm for the Reward Maximization Problem With Non-Deterministic Rewards

We now consider the reward maximization problem with non-deterministic rewards, given a set R of non-preemptive AR requests. We first propose an approximation algorithm for a special case of the problem, assuming that all tasks of each request $r_j \in R$ can be consolidated into a single base station. We then extend the proposed approximation algorithm for the case without the assumption.

TABLE I
SYMBOLS

Symbols	Meaning
$G = (\mathcal{BS}, E)$	an MEC network with a set $\mathcal{BS}$ of base stations and a set E of links/paths that interconnect the base stations.
$i \in \mathcal{BS}$, $e \in E$, $C(i)$	a base station, a link in E, and the computing capacity on base station i.
r_j	a request of an AR application.
$\{(j,1), \cdots (j,k), \cdots (j,K_j)\}$	the sequence of tasks of r_j with $1 \leq k \leq K_j$.
ρ_j and $\mathcal{DR}$	the data rate of video streams of AR request r_j and a finite set of possible data rates that each AR request may have.
S_j	the service required by request r_j.
$\pi_{j,\rho}$ and $RD_{j,\rho}$	the probability of AR request r_j with a data rate ρ and the corresponding reward.
a_j and b_j	the arrival time slot of request r_j and the time slot that the first frame of r_j's video stream is processed.
ρ_{unit}	the minimum size of video frames that is needed to perform each augmentation.
C_{unit}	the amount of computing resource needed to process a unit data rate.
d_{jki}^{pro}	the processing delay of ρ_{unit} amount of data by task (j,k) of r_j in base station i.
d_{je}^{trans}	the transmitting delay of ρ_{unit} along link e in the MEC network G.
D_j and $\hat{D}_j$	the experienced latency and the latency requirement of AR request r_j.
x_{ji}	a binary indicator that shows whether tasks of request r_j are assigned to base station i for implementation.
p_{ji}	the shortest path between the mobile user of r_j and base station i in G in terms of transmission latency.
$E(\rho_j)$	the expected size of AR request r_j.
$C_{l,i}$ and L	the capacity of resource slot l in base station i and the number of resource slots.
y_{jil}	a binary indicator that shows whether request r_j is assigned to the lth resource slot of base station i.
ER_{jil}	the expected reward of request r_j assigned to the lth resource slot of base station i.
$R_{l,i}$	the number of requests that are randomly assigned to resource slot l of base station i.
R^{Pre}	the set of requests that are pre-assigned to basestations of the MEC network G.
$LPOpt$, Opt	the optimal solutions to **LP** and the reward maximization problem with non-preemptive AR requests, respectively.
C^{th} and $Z = [C_{min}^{th}, C_{max}^{th}]$	the threshold of the minimum amount of computing resource assigned to each request at each time slot and the range of C^{th}.
$ER(a)$, η and T	the expected reward of selecting an arm a in Z, a given constant in Eq. (12), and the entire time horizon.
κ and ϵ	the number of intervals that Z is discretized and the fixed length of subintervals.
Z'	a set of finite arms after the discretization of Z.
$ER^*(Z')$	the best arm in Z' that achieves the maximum reward of scheduling the requests.
$UCB_t(a)$, $LCB_t(a)$, $r_t(a)$	the upper and lower bounds on the reward of each arm a exist in each time slot t, and the confidence radiation.
R_t and $\lvert R_t \rvert$	a set of requests to be scheduled for implementation at time slot t and the number of requests in R_t.
ρ_{max} and ρ_{min}	the maximum and minimum of predicted data rates of requests.
$C_{i,l}$	the value of the size of resource slots on base station i.
H and h	the number of groups needed for the MT-LSTM of each learning task and the index of each group.
u_t^h	an LSTM unit in group h at time slot t.
Φ and F	the number of AR application types, and the average data rate of all implemented requests so far.
s_i and WT_j	a segment of requests divided by data rate, and the waiting time of request r_j.
w_j and W_{s_i}	the weight of request r_j and the weight of segment s_i.
μ	a parameter that captures the relation between the remaining time and data rate of requests.
W	the number of edges in the neural network of MT^2-LSTM.

A. Approximation Algorithm for a Special Case of the Problem With Consolidated Tasks

Assuming that all tasks of each request r_j can be consolidated into a single base station, the basic idea of our algorithm is to propose an exact solution to the problem by formulating an Integer Linear Program (ILP) first. We then develop a randomized algorithm for the problem base don the linear relaxation on the ILP.

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

$$D_j = b_j - a_j + \sum_{i \in \mathcal{BS}} x_{ji} \Big(\sum_{e \in p_{ji}} 2 \cdot d_{je}^{trans} + \sum_k d_{jki}^{pro} \Big), \quad (1)$$

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

Recall that the objective of the reward maximization problem with a set of non-preemptive AR requests is to maximize the expected total reward of admitted requests. Denote by $E(\rho_j)$ the expected size of request r_j, i.e., $E(\rho_j) = \sum_{\rho \in \mathcal{DR}} \pi_{j,\rho} \cdot \rho$. We formulate this special case of the problem with all tasks of each request being consolidated into a single base station in the following.

ILP-RM:

$$\max \sum_j \sum_i x_{ji} \cdot \sum_{\rho \in \mathcal{DR}} \pi_{j,\rho} \cdot RD_{j,\rho},$$

$$\text{subject to, } \sum_i x_{ji} \leq 1, \quad \forall r_j \quad (2)$$

$$\sum_j x_{ji} \cdot E(\rho_j) \cdot C_{unit} \leq C(i), \quad \text{for } i \in \mathcal{BS} \quad (3)$$

$$D_j \leq \hat{D}_j, \quad \text{for each } r_j \quad (4)$$

$$x_{ji} \in \{0,1\}, \quad (5)$$

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 base station i. This may be a feasible solution, given deterministic rewards. However, the objective of the reward maximization problem is to maximize the expected total reward of admitted requests. If we follow the mentioned simple relaxation, each request may instantiate to a higher data rate to gain a higher

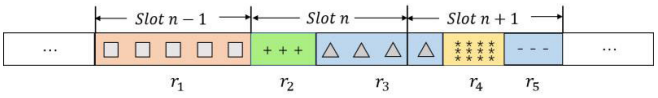


Fig. 2. Requests placed in different computing resource slots of i.

expected reward. This creates contention in each base station that prefers to admit requests with larger data rates. However, the probability of requests with large data rates is usually small [15]. The expected total reward obtained then is very likely to be sub-optimal.

We observe that the computing resource capacity $C(i)$ is enforced for each base station 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**, following similar resource allocation in cloud computing [33], [56]. Specifically, we partition the capacity $C(i)$ of base station 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 $C_{l,i}$ the capacity of resource slot l in base station i. The resource capacity of base station i thus is divided into $L = \lfloor C(i)/C_{l,i} \rfloor$ resource slots, and the numbers of resource slots of all base stations are the same. The resource slots of each base station are indexed by l, and assigned to requests one by one according to the indices. Namely, if request r_j is assigned a *resource slot* starting at l, it means that the computing resource assigned to r_j starts with resource slot l of base station i. It however may span multiple resource slots of base station i starting from the lth resource slot. The reason is that the amount of computing resource allocated at resource slot l may be insufficient 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 resource-slotted base station.

We describe the LP relaxation on the **ILP-RM**. Variable y_{jil} is a binary indicator that shows whether request r_j is assigned to the lth resource slot of base station i or not. Then,

$$x_{ji} = \sum\nolimits_l y_{jil}. \tag{6}$$

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

$$ER_{jil} = \sum_{\rho \in \mathcal{DR}: \rho \cdot C_{unit} \le C(i) - l \cdot C_{l,i}} \pi_{j,\rho} \cdot RD_{j,\rho}. \tag{7}$$

Eq. (7) shows 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**, can be written as

LP:

$$\max \sum\nolimits_j \sum\nolimits_i \sum\nolimits_l y_{jil} \cdot ER_{jil},$$

$$\text{subject to } \sum\nolimits_i \sum\nolimits_l y_{jil} \le 1, \forall r_j \tag{8}$$

$$\sum_j \sum_{l':l' \le l} y_{jil'} E(\min\{\rho_j, \frac{l \cdot C_{l,i}}{C_{unit}}\}) \le 2 \frac{l \cdot C_{l,i}}{C_{unit}}, \ \forall l, i \tag{9}$$

$$D_j \le \hat{D}_j, \ \forall r_j \tag{10}$$

$$0 \le y_{jil} \le 1, \tag{11}$$

where Constraint (8) indicates that each request can only be assigned to a resource slot of a base station. Constraint (9) includes a down-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 is to consider the fact that a request may have a large data rate with low probability, if assigning such requests to the resource slots may make the requests with low data rates and high rewards being rejected. Thus, Constraint (9) avoids the contention of assigning requests with large data rates and low probabilities to the first $2 \cdot l$ resource slots. Constraint (10) is the delay requirement, where D_j is calculated by substituting x_{ji} in Eq. (1) with $\sum_l y_{jil}$. Constraint (11) says that y_{jil} is a real value in range $[0, 1]$.

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 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 $i \in \mathcal{BS}$ **do**
5: Consider the request with the lth smallest data rate;
6: **if** the requests assigned so far in base station i occupy at most $l \cdot C_{l,i}$ amount of computing resource **then**
7: Assign r_j to base station i;

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

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. We here remove this assumption by proposing an efficient heuristic for the problem.

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, 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 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 $i \in \mathcal{BS}$ **do**
6: Consider the request r_j with the lth smallest data rate;
7: **if** the requests assigned so far in base station i occupy at most $l \cdot C_{l,i}$ amount of computing resource **then**
8: Pre-assign r_j to base station i;
9: $R^{Pre} \leftarrow R^{Pre} \cup \{r_j\}$;
10: **else**
11: For the requests that are already pre-assigned to base station 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 $l \cdot C_{l,i}$ **then**
13: Pre-assign a task of $r_{j'}$ to the closest base station i;
14: Pre-assign r_j to base station i;
15: $R^{Pre} \leftarrow R^{Pre} \cup \{r_j\}$;
16: Assign the tasks of requests according to the pre-assignment in R^{pre};

Our basic idea is to perform a 'pre-assignment' according to algorithm Appro. The obtained solution will be considered as the final assignment if no requests are rejected due to Step 6. Otherwise, we adjust the 'pre-assignment' by avoiding the migration of some tasks that belong to the already 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 the request. Specifically, let R^{Pre} be the set of requests that are pre-assigned to base stations. Each request is pre-assigned to G, following 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 $l \cdot C_{l,i}$, 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 the **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 $LPOpt \geq Opt$.

Please see the proof body of Lemma 1 in the Appendix.

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 $\frac{1}{2}$.

Please see the proof body of Lemma 2 in the Appendix.

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

Please see the proof body of Theorem 1 in the Appendix.

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 implementation, there is an efficient heuristic algorithm, Heu, for the reward maximization problem that delivers a feasible solution.

Please see the proof body of Theorem 2 in the Appendix.

V. Online Learning Algorithm for the Dynamic Reward Maximization Problem With Non-Deterministic Rewards

A. Online Learning via Lipschitz Bandits

To meet the latency requirement of each request, a naive approach is to schedule requests immediately after their arrivals, such that each request is responded immediately. However, the system may not have sufficient computing resource to meet the total resource demand of all requests. We may allow equal resource sharing among the requests within each time slot. If there is a burst of requests to be scheduled in a particular time slot, the base stations will be overloaded if all arrived requests equally share the resource at a base station. 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 are infinite numbers of values in the range of $Z = [C^{th}_{min}, C^{th}_{max}]$. To identify a proper value of C^{th}, the basic idea is to discretize the continuous range into a finite set of candidate values, 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 as follows.

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:

$$|ER(a) - ER(b)| \leq \eta |a - b|, \text{for any arms } a, b \in Z, \quad (12)$$

where η is a given constant.

We perform discretization on the value interval Z by dividing it into κ subintervals with fixed length $\epsilon = \frac{C^{th}_{max} - C^{th}_{min}}{\kappa - 1}$, 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 maximize the expected reward $ER^*(Z')$. To this end, we adopt a successive elimination algorithm. That is, all arms in Z' are set to "active" initially. In each time slot t, we assume that there

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, 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 κ intervals with fixed length $\epsilon = \frac{C^{th}_{max} - C^{th}_{min}}{\kappa - 1}$;
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**;

are an upper bound and a lower bound on the reward of each arm a. 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 one arm left in Z', which is selected as the threshold C^{th}_t at time slot t.

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

LP-PT:

$$\max \sum_j \sum_i \sum_l y_{jil} \cdot ER_{jil},$$

subject to constraints (10), (11), and

$$\sum_i \sum_l y_{jil} \leq 1, \forall r_j \in R_t \tag{13}$$

$$\sum_j \sum_{l':l' \leq l} y_{jil'} E(\min\{C(i)/|R_t|, \rho_j, l \cdot C_{l,i}/C_{unit}\}) \leq 2 \cdot l \cdot C_{l,i}/C_{unit}, \forall l, i \tag{14}$$

where $|R_t|$ is the number of requests to be scheduled for implementation 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 the expected data rate. We then add requests to R_t one by one. Note that a larger set of R_t usually means less resource is allocated to each request. We keep adding requests to R_t. This however makes each request being assigned to a smaller amount of resource. Therefore, such addition of requests continues until 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 are given in **Algorithm** 3, which is referred to as `DynamicRR`.

B. Algorithm Analysis

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, $\epsilon = \frac{C^{th}_{max} - C^{th}_{min}}{\kappa - 1}$, T is the entire time horizon, and η is the given constant of the Lipschitz condition in Eq. (12).

Please see the proof body of Theorem 3 in the Appendix.

Remark: Note that the regret can be simplified to $O(\sqrt{T \log T/\epsilon} + T \cdot \epsilon)$. In other words, the value of ϵ determines the number of intervals, and more intervals means that the obtained threshold is more accurate. In this way, the threshold may be much more appropriate to the dynamics and uncertainty of the MEC network.

VI. Fairness-Aware Online Learning Algorithm for the Dynamic Reward Maximization Problem With Non-Deterministic Rewards via Data Rate Prediction

We now propose a machine learning-based algorithm for coping with fair admissions of requests for the dynamic reward maximization problem with non-deterministic rewards.

A. Overview

The proposed algorithms `Appro`, `Heu`, and `DynamicRR` may lead to the 'unfair' scheduling of requests with high data rates and low rewards, due to the following reasons. First, algorithms `Appro` and `Heu` use resource-slot-indexed relaxation of the ILP to find feasible solutions, by dividing base stations into resource slots with the same size. In particular, to avoid reward reduction due to admissions of requests with large data rates, we limit the size of each resource slot and requests that can be assigned to the resource slot, as shown in Constraint (9). Second, algorithm `DynamicRR` limits the resource allocation to a request, such that future requests can be responded with sufficient resource. The aforementioned unfair scheduling may lead to a higher response latency and violation of latency requirements. In particular, if the resource demand of some requests are higher than the size of the resource slot and the threshold, the requests have to wait for being scheduled. Consequently, the requests with high data rates and low rewards may be scheduled in low priority. In this extreme case, they may wait for a very long time for admissions. In practical applications, considering a request with surge of data rate waiting for timely processing, its implemention may not lead to a high reward due to the cost of much resource consumption. However, if the request is not assigned with a higher priority, its latency requirement will be violated, thereby downgrading the quality of services.

To avoid the aforementioned two issues, we allow each base station i to have a different value of the size of its resource slots, denoted by $C_{i,l}$. Only requests with data rates close to $C_{i,l}$ can be assigned to the base station. Besides, we adopt a novel prediction mechanism to accurately predict the data rates of requests. The basic steps of the proposed heuristic consists of three stages: **Stage 1**: predicting the data rates of requests, **Stage 2**: dividing the data rates of requests into a number of segments, and **Stage 3**: request assignment to base stations.

B. Algorithm

We now describe the three stages of the online learning algorithm.

Stage 1: Predicting data rates. We observe that the data volumes of AR requests have a strong temporal pattern, depending on the most recent events or some events happened in a longer time scale of the AR application. For example, in an AR application, the data rate of its request depends on the objects placed in the most recent scenes of the physical environment. That is, in an online shopping AR application, more requests may be sent if the current shopping area is of interest of buyers. In addition, different AR requests have different temporal patterns of the data volumes. The prediction has to consider various temporal patterns of data volumes.

The multi-timescale long short-term memory neural network (MT-LSTM) allows the long short-term memory neural network (LSTM) to learn time series with patterns that span different time scales [32]. An MT-LSTM learns when to forget the historical information of the memory unit, when to update the memory unit with new information. To learn across different time scales, it partitions the gates, memory cells, and hidden states into H groups with each group being activated at different time scales [32]. However, the MT-LSTM is designed for natural language processing applications, and directly application of the MT-LSTM may not capture the dynamics of IoT applications. Besides, there are multiple types of AR applications with different data rates. Learning the data rates of one type of AR application may provide insights for other types. By adopting the MT-LSTM, we could obtain accurate data rates for each type of AR applications. This however may not be efficient in terms of convergence speed. An alternative is to adopt the multi-task learning with the prediction of each type of AR applications being represented as a learning task. Multi-task learning learns the correlation between multiple tasks, which integrates multiple training models into one model to share the parameters learned from different tasks.

Based on the mentioned observation, we design a novel neural network based on multi-task and multi-timescale LSTM (MT^2-LSTM) learning. Specifically, assume that there are Φ types of AR applications, we create a learning task for each of the Φ types of AR applications. Each learning task adopts an MT-LSTM network to predict the data rates of the requests of its corresponding AR application. To this end, we consider that each time scale consists of a number of time slots. We then determine the number H of groups needed for the MT-LSTM of each learning task. Intuitively, larger groups imply that the temporal pattern of data rates of requests spans longer time scales. Since such information is not known before the request arrives, we use the average data rate of requests implemented so far. Let F be the average data rate of all implemented requests so far. The maximum number of groups for the prediction of the total data volume of each request then is $H = \log_2 F - 1$.

For each learning task, we introduce the architecture of the model, which consists of activated and inactive units and the transitions between units. The LSTM units of the model are partitioned into H groups, and each group is indexed by h, i.e., $1 \leq h \leq H$. In particular, u_t^h is defined as an LSTM unit in group h at time slot t. As shown in Figure 3, the red-border nodes represent activated LSTM units,

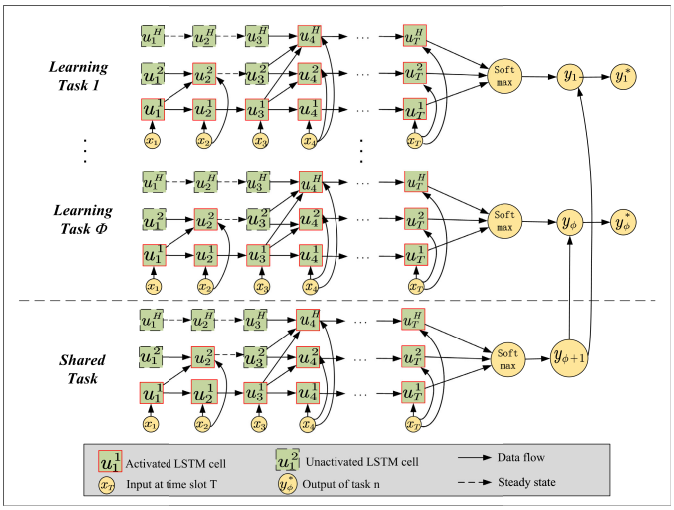


Fig. 3. The architecture of MT^2-LSTM model.

while black-dotted-border nodes represent LSTM units that are currently inactive. The black-solid lines represent the units that will be updated at the next time slot, while the black-dotted lines represent the units which remain unchanged. The LSTM units of each group h are activated at different time periods T_h. In this approach, we adopt an exponential series of periods: group h has the period of $T_h = 2^{h-1}$. When group h is activated at time slot t ($1 \leq t \leq T$), the LSTM units in this group are updated based on the input of the current time slot and the output of the previous time slot. When group h is inactivated at time slot t, its LSTM units remain unchanged. At the end of each hidden layer, there is a fully connected layer followed by a softmax nonlinear layer that predicts the probability distribution of the data rate of the AR request.

We also enable the sharing of parameters among different learning tasks, via experience sharing among all the Φ learning tasks, by adding a shared MT-LSTM model to learn the common experience from different AR applications [27] shown in the bottom of Fig. 3. To learn the data rates of requests of a type of AR application, its historical information will be fed into both of its learning task and the learning task with the shared MT-LSTM. Note that after getting the predicted data rates, the reward of implementing each request is obtained, since the data rate determines the amount of reward received.

Stage 2: Dividing the data rate into a number of segments. Given the predicted data rates of requests, we obtain the maximum and minimum data rates, denoted by ρ_{max} and ρ_{min}, respectively. We divide the range $[\rho_{min}, \rho_{max}]$ to $|\mathcal{BS}|$ segments. Each base station $i \in \mathcal{BS}$ corresponds to a segment $[\rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot (i-1), \rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot i)$. We set the value of the resource slot of base station i to the median of the length of its corresponding segment, i.e.,

$$C_{i,l} = \rho_{min} + (\rho_{max} - \rho_{min}) \cdot (i-1))/(2 \cdot |\mathcal{BS}|). \quad (15)$$

It must be mentioned that the segments may have unbalanced numbers of requests in their range. Meanwhile, some requests have been waiting for a long time, with higher priorities to be processed. Thus, we set a weight for each segment. The

requests in the segment with a larger weight will be assigned earlier (with a higher probability).

To set the weight of each segment, we follow a customized method of the well-known highest response ratio scheduling policy in modern operating systems. Specifically, the response ratio of a request is the ratio of

$$\frac{waiting\ time\ +\ remaining\ time}{remaining\ time}. \tag{16}$$

We however do not know the remaining execution time of each request. Instead, we assume that the remaining execution time is proportional to its data rate. As a result, the weight of a request is

$$w_j = (WT_j + \mu \cdot \rho_j)/(\mu \cdot \rho_j), \tag{17}$$

where WT_j is the waiting time of request r_j, $\mu \cdot \rho_j$ is used to approximate the remaining execution time and μ is a parameter to capture the relationship between the remaining execution time and the data rate of the request. The weight W_{s_i} of segment s_i thus is

$$W_{s_i} = \sum\nolimits_{r_j \in s_i} w_j. \tag{18}$$

Stage 3: Assigning requests to base stations. We assign requests with different segments of data rates into base stations in $\mathcal{BS}$. Recall that in **Stage 2**, we divide the values of data rates into $\mathcal{BS}$ segments. The set of requests in segment s_i consists of requests with data rates in the range of $[\rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot (i-1), \rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot i)$. We use `Heu` to assign the requests of each segment to base stations.

To fairly assign requests to base stations, we adopt an iterative assignment policy. In each iteration, we select at most one request from each segment according to the weight of the segment. In more detail, the probability of selecting a request from R_{s_i} is

$$W_{s_i}/(\sum\nolimits_{s_i} W_{s_i}), \tag{19}$$

Given the selected requests from all segments, we invoke algorithm `Heu`. However, algorithm `Heu` adopts a uniform resource slot length for all base stations. We modify algorithm `Heu` to consider different lengths of resource slots by modifying Constraint (9) to

$$\sum\nolimits_j \sum\nolimits_{l':l' \le l} y_{jil'} E(\min\{\rho_j, \frac{l \cdot C_{i,l}}{C_{unit}}\}) \le \frac{2l \cdot C_{i,l}}{C_{unit}}, \quad \forall l, i \tag{20}$$

This procedure of request assignment continues until no more requests can be assigned. The detailed steps of the proposed algorithm are shown in **Algorithm** 4.

Theorem 4: Algorithm `DynamicFair` delivers a feasible solution to the reward maximization problem in time $O(W+|R|^2 \cdot |\mathcal{BS}|+|R|(n^\omega+n^{2.5-\alpha/2}+n^{2+1/6})I\log^k((n^\omega+n^{2.5-\alpha/2}+n^{2+1/6})I))$, where $n = |R| \cdot |\mathcal{BS}| \cdot L$ and W is the number of edges in the neural network of MT^2-LSTM.

Please see the proof body of Theorem 4 in the Appendix.

VII. Experiments

The rest is to study the performance of the proposed algorithms in a real test bed.

Algorithm 4 A Fairness-Aware Online Learning Algorithm for the Dynamic Reward Maximization Problem, i.e., `DynamicFair`

Input: An MEC network $G = (\mathcal{BS}, E)$, a set of requests, and 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: **for** each time slot $t \leftarrow 1 \cdots T$ **do**
2: Predicting the data rates of requests in time slot t by the MT^2-LSTM prediction method;
3: Dividing the predicted values of data rates into $|\mathcal{BS}|$ segments, with each segment consists data rates in the range of $[\rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot (i-1), \rho_{min} + \frac{\rho_{max}-\rho_{min}}{|\mathcal{BS}|} \cdot i)$;
4: Calculate the size of computing slots on each base station by Eq. (15);
5: **for** $iter \leftarrow 1, \cdots, ITER$ **do**
6: Update the weight of each segment. Set the weight to 0 if there is no request remained in the segment, else obtain the weight of each segment by Eq. (17) and Eq. (18).
7: Select at most one request in each segment according to the probability of R_{s_i} by Eq. (19), and add it to the set of currently-considering set of requests;
8: Invoke algorithm `Heu` with Constraint (20), and the currently-considering set of requests as its input;

(a) The hardware platform (b) Network Topology

Fig. 4. A testbed with hardware platform and network topolopy.

A. Testbed Settings

Our testbed consists of 11 heterogeneous devices, 6 of them serving as the users of the system, which send their AR requests to the MEC network. As shown in Fig. 4 (a) and (b), user equipments include two NVIDIA Jetson TX2, three NVIDIA Jetson AGX Xavier, and one desktop PC equipped with an i7-10700F CPU and 16GB of RAM used to generate AR requests. In the MEC network, we use three NVIDIA Jetson Nano serving as base stations. The computing resources of each base station thus is set to 1430MHz. The capacity of each resource slot is 700 MHz [58] with each base station having around two resource slots. We also have a desktop PC with an i7-10700F CPU and 16GB of RAM serving as a hypervisor that runs the proposed algorithms to decide the offloading and scheduling of AR tasks. All user equipments and base stations are interconnected by a Huawei S5720-32C-HI-24S-AC switch. The bandwidth between a user and a cloudlet is randomly drawn from [80,120] Mbps [37].

We recorded 300 30-second AR videos of different sizes with special markers, and input all the video into two marker-based AR applications separately. Then, we tracked the data rate of each video every 30 frames when an AR applications process AR requests, and data show that the data rate of each request is varied from [4,36] Mbps. Each

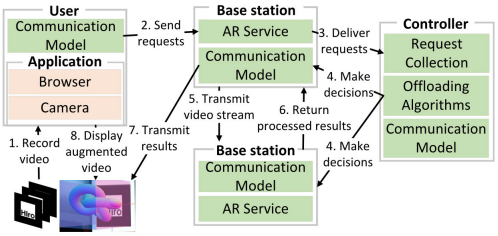


Fig. 5. The software platform.

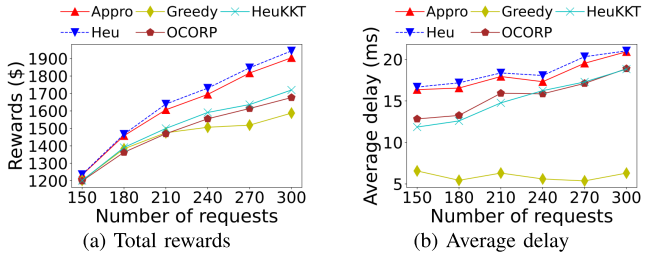


Fig. 6. The impact of the number of requests on the performance of algorithms Appro, Heu, Greedy, OCORP and HeuKKT.

request has 4 tasks. Processing AR requests requires 50 MHz of computing resource per MB for each base station [49]. The number of AR requests varies from [150, 300]. The delay requirement each AR request is 200 milliseconds [7]. The reward for completing a 30-second AR request is in the range of [5,12] dollars [43]. The length of each time slot is 333 milliseconds [58].

We use 200 videos along with their corresponding data rates as the training set to train the MT^2-LSTM model, remaining 100 videos as the test set. We initialized the weight parameters by randomly sampling from uniform distribution in [−0.1,0.1] [32]. We then trained the MT^2-LSTM model, using forward propagation and back propagation similar to LSTM, but with one difference, inactive nodes retain the state of the previous round. We then optimize based on gradient using the Adagrad update rule [32]. Our goal is to minimize the cross-entropy of the predicted distribution and the true distribution of the data rates of the AR video stream. Finally, we repeat the training and choose hyper parameters that achieve the best performance on the test set.

B. Software and Benchmarks

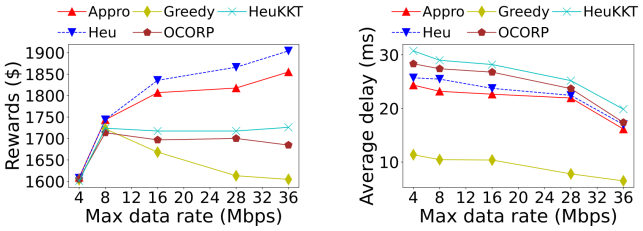
We developed a software system to process video streams of AR requests, as shown in Fig 5. We developed two marker-based AR applications using 'AR.js', which is a lightweight JavaScript library for AR on the web [34], [41], [53]. We then deployed web AR applications in user equipments to allow users to send AR requests. The workflow of the software system consists of seven steps: recording video, sending AR requests, delivering information, making offloading decisions, transmitting video stream, returning processed results and displaying augmented video. Specifically, in step 1, we record a total of 75 videos, with resolution options of 540p, 720p, and 1080p. In step 2, users issue requests by accessing the URL of specific service deployed in the nearest base station. Next, users start to transmit video streams by building HTTP connections with base stations. In step 3, when a base station receives requests, it extracts the information of the video stream, and forwards it to the controller. In step 4, the controller invokes different algorithms to make offloading decisions with the collected information of requests, and sends the decision of each request to corresponding base stations. The video streams are then forwarded according to the selected base station in step 5. In step 6, the results are sent back to each user equipment. In step 7, the web browser of each user equipment displays the augmented video stream.

We evaluated the proposed algorithms against the following benchmark algorithms: (1) The first benchmark is based on an online convex optimization in [35], which is referred to as OCORP. Specifically, in each time slot, OCORP sorts requests by their arriving time and remaining to-be-processed data, then assigns requests to edge servers based on a best-fit algorithm. (2) The second benchmark is a greedy algorithm [60], referred to as Greedy, which greedily assigns AR requests with potentially large data rates the task to the optimal edge server one-by-one. (3) The third benchmark is the heuristic in [40], referred to as HeuKKT, which considers service caching and task scheduling by fomulating a non-linear program. The proposed solution divides the original algorithm into two subproblems by Karush-Kuhn-Tucker conditions.

C. Performance of Appro and Heu

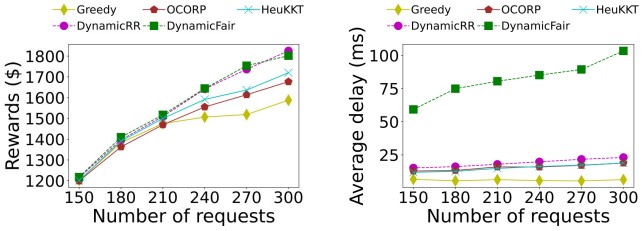
We first studied the performance of algorithms Appro, Heu, OCORP, Greedy, and HeuKKT in terms of the total reward and average delay of non-preemptive AR requests, by varying the number of requests from 150 to 300. From Fig. 6(a), we can see that algorithm Appro achieves 13.6%, 19.8%, 10.7% larger rewards than those of algorithms OCORP, Greedy, and HeuKKT when the number of requests is 300. Also, algorithm Heu has 15.9%, 22.4%, and 13.0% larger rewards than those of algorithms OCORP, Greedy, and HeuKKT, respectively. The reason is that algorithms Appro and Heu deal with the uncertainty of data rates of requests carefully, by considering a resource slot based allocation. Instead, algorithms OCORP, Greedy, and HeuKKT adopt coarse-grained methods according to the data rates of requests, implementation times, and the resource capacity constraints of edge servers. Additionally, the obtained rewards by all comparison algorithms first increase rapidly and then rise slowly, with the growth of the number of requests. This is due to that the network resource becomes saturated and no more requests can be admitted, with the growth on the number of requests. From Fig. 6 (b), we can see that the average delays of algorithms OCORP, Greedy and HeuKKT are lower than those of the rest algorithms, because they trade-off the rewards for delays. Although Appro and Heu have higher delays, they have much larger rewards than the rest algorithms.

We then investigated the performance of algorithms Appro, Heu, OCORP, Greedy, and HeuKKT in the terms of the total reward and average delay of 300 non-preemptive requests, by varying the maximum data rate of requests from 4 to 36 Mbps. From Fig. 7 (a), we can see that the



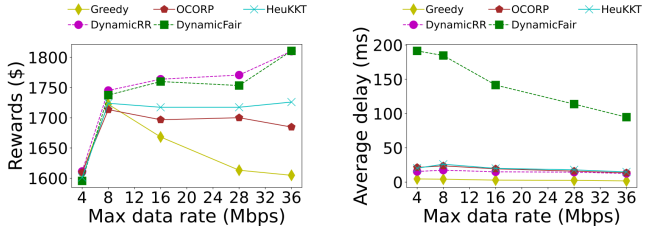
(a) Total rewards (b) Average delay

Fig. 7. The impact of maximum data rate of requests on theThe performance of algorithms Appro, Heu, Greedy, OCORP and HeuKKT.



(a) Total rewards (b) Average delay

Fig. 8. The impact of the number of requests on the performance of algorithms DynamicRR, DynamicFair, Greedy, OCORP and HeuKKT.



(a) Total rewards (b) Average delay

Fig. 9. The impact of maximum data rate of requests on the performance of algorithms DynamicRR, DynamicFair, Greedy, OCORP and HeuKKT.

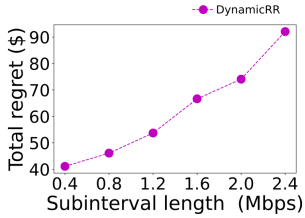


Fig. 10. The regret of DynamicRR by varing subinterval length ϵ.

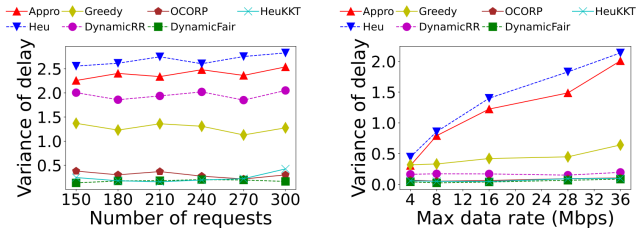
rewards rapidly increase when the maximum data rate is in the range of [4, 8] Mbps for all algorithms. The rewards of algorithms Appro and Heu grow slowly, while the rewards of algorithms HeuKKT and OCORP keep stable and the rewards of algorithm Greedy decrease, when the maximum data rate increases from 8 to 36 Mbps. The rationale behind is that requests with larger data rates require more computing resources, and they may have fewer rewards. The benchmark algorithms give requests with large data rates with higher priorities, so that the total number of completed requests decreases and the total reward does not increase or even decreases. In addition, Fig. 7(b) shows that the average delay of all algorithms decreases with the increase on the maximum data rate. Also, algorithm Greedy achieves the minimum average delay. The reason is that an increase in the maximum data rate leads to a decrease in the number of requests processed per time slot, resulting in a decrease in the average delay of the requests.

D. Performance of DynamicRR *and* DynamicFair

We then evaluated the performance of algorithms DynamicRR, DynamicFair, OCORP, Greedy and HeuKKT in terms of the total reward and average delay of preemptive AR requests, by varying the number of requests from 150 to 300. It can be seen from Fig. 8 (a) that algorithms DynamicRR and DynamicFair have much larger rewards than those of algorithms OCORP, Greedy and HeuKKT, respectively. This is because DynamicRR adopts a dynamic threshold adjusting method to avoid starvation of AR requests with low rewards, and DynamicFair accurately predicts the data rates of AR requests. As shown in Fig. 8 (b), algorithm DynamicFair has a higher average delay than those of the other algorithms, and the average delay of all algorithms increases with the growth of the number of requests. The reason is that DynamicFair trade-offs delay for fairness.

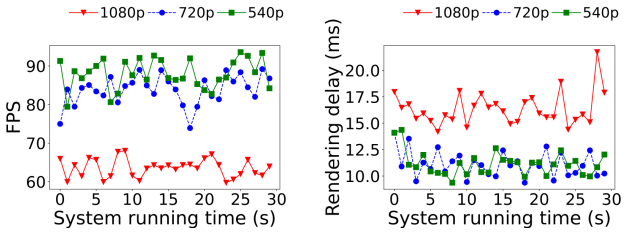
We then investigated the impact of the maximum data rate of preemptive AR requests on the performance of algorithms DynamicRR, DynamicFair, OCORP, Greedy and HeuKKT, by varying the maximum data rate of 300 requests from 4 to 36 Mbps. From Fig. 9 (a), we can see that the rewards of algorithms DynamicRR and DynamicFair grow slowly as the maximum data rate increases. All comparison algorithms grow in rewards when the maximum data rate is in the range of [4,8] Mbps, but OCORP and HeuKKT remain stable, while Greedy decreases as the data rate increases further. The reason is that our algorithms can predict the data rate of each request and divide the resource slots for fine-grained offloading. In contrast, the benchmark algorithms focus more on requests with large data rates and spend more computing resources, but they potentially earn fewer rewards. As shown in Fig. 9 (b), the average delay of algorithm DynamicFair is substantially larger than the other algorithms. Also, its average delay decreases rapidly as the maximum data rate increases, while the average delays of the other algorithms keep stable. The reason is that the number of requests decreases with the growth of maximum data rate, resulting in a decrease of total predicting delay for algorithm DynamicFair.

We also studied the impact of the length ϵ of subintervals on the regret of algorithm DynamicRR, by dividing 300 requests with data rates in the range of 4 to 36 Mbps into subintervals with their lengths varied from 0.4 to 2.4 Mbps. As shown in Fig. 10, we can see that the regret of DynamicRR gradually increases when the length ϵ of subintervals increases from 0.4 to 2.4 Mbps. The reason is that an increase of the length ϵ of subintervals results in a decrease in the number of subintervals; at the same time, the value of threshold C^{th} is chosen from a smaller value pool and leading to a higher distance from the optimal threshold. Consequently, this may not adapt better to the dynamics and uncertainty of the MEC network. This trend can be verified by the obtained regret $O(\sqrt{T \log T/\epsilon} + T \cdot \epsilon)$.



(a) Fairness of algorithms (b) Fairness of algorithms

Fig. 11. The fairness of algorithms Appro, Heu, Greedy, DynamicRR, DynamicFair, OCORP and HeuKKT, by varing the number and maximum data rate of requests.



(a) Frames per second (b) Rendering delay

Fig. 12. The impact of the resolution.

We now use the variance of delay to measure the fairness among algorithms DynamicFair, DynamicRR, Appro, Heu, OCORP, Greedy and HeuKKT, by varying the number of requests from 150 to 300 and the maximum data rate of requests from 4 to 36 Mbps. As shown in Fig. 11 (a) and (b), we can see that DynamicFair has the lowest variance, indicating that DynamicFair is the fairest algorithm for users, while the variance of Heu and Appro are higher than any other algorithms. The reason is that DynamicFair fairly treats requests to avoid the starvation of some requests. Algorithms Appro and Heu consider non-preemptive requests, leading to a larger gap in waiting time between requests compared to the rest algorithms. Further, as shown in Fig. 11 (a), the variance of delay for all algorithms keeps stable as the number of requests increases, because the delay of all requests grows with the increase in the number of requests resulting in a smaller gap between different requests. Fig. 11 (b) shows the variance of all requests increases with the rise of data rate, because requests with different data rates have various delays.

E. System Performance

We finally evaluated the system performance in the test-bed in the terms of the completion rate of requests, FPS, and the rendering delay. As shown in Table II, we can see that all algorithms achieve a completion rate of over 80% for a range of concurrent numbers from 30 to 54. The reason is that our system can make fully use of the computing resources of the base stations to process more requests. However, due to the limitations of computing resources, the completion rate decreases as the concurrent number increases. Fig. 12 (a) and (b) show that our test-bed achieves more than 60 FPS and less than 20 milliseconds rendering delay for augmented video in 540p, 720p, and 1080p. Additionally, the FPS of augmented video in 540p and 720p is higher than that of 1080p. Besides, the rendering delay of augmented video in 540p and 720p is lower than that of 1080p. The rationale behind is that requests with 540p and 720p resolutions have lower data rates and require fewer computing resources, while those with 1080p resolution require more resource to render.

TABLE II
REQUEST COMPLETION RATE WITH DIFFERENT CONCURRENCY

Concurrency	30	36	42	48	54
Appro	100%	98.7%	94%	89.3%	86.8%
Heu	100%	99.2%	96.2%	90.5%	86.5%
DynamicRR	99.1%	93.5%	89%	88%	83.7%
DynamicFair	98.2%	95.3%	87.1%	84.5%	82.2%

VIII. CONCLUSION

In this paper, we studied the reward maximization problem of 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 algorithms for the dynamic reward maximization problem if AR requests are allowed to be preemptively scheduled, by leveraging the multi-armed bandit technique, and we also proposed a fairness-aware online algorithm for the problem to avoid starvations of AR requests, by adopting a novel MT^2-LSTM method to predict data rates of requests. We finally evaluated the performance of the proposed algorithms by emulations in a real test-bed. Experimental results show that the proposed algorithms outperform that of existing studies by improving the expected total reward by no less than 13%.

APPENDIX

Proof of Lemma 1

Proof: We show that Opt is the optimal solution to **LP**. To this end, we need to show that solution Opt meets constraints (8), (9), and (10), respectively. It can be seen that Opt meets constraints (8) and (10) because $x_{ji} = \sum_l y_{jil}$. We thus need to show that Opt also meets constraint (9). Specifically, we consider 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. Let x^*_{jil} and $s^*_{j,\rho}$ be binary variables that indicate whether request r_j is assigned to the resource slot l of bs_i in the optimal solution Opt and r_j has size of ρ, respectively.

Let X_l be the request that is assigned to the resource slot that is right before resource slot l. Note that request X_l occupies a continuous amount of computing resource of bs_i, instead of scattered resource slots. Thus, request X_l is the only request in Opt that its demanded computing resource may exceed that of its assigned resource slot. This means that

$$\sum_{r_j \neq X_l} \sum_{\substack{l' \leq l \\ \rho \cdot C_{unit} \leq C(bs_i)}} x^*_{jil} \cdot s^*_{j,\rho} \cdot \rho \leq \frac{l \cdot C_l}{C_{unit}}. \quad (21)$$

If we include request X_l in Ineq. (21) and truncating the data rate by $\frac{l \cdot C_l}{C_{unit}}$, we then have

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

Recall that we consider algorithms that schedule a request before knowing the data rate of the request. Since both x^*_{jil} and $s^*_{j,\rho}$ are independent variables, we re-write the LHS of Ineq. (22) as

$$\begin{aligned}
&\sum_{r_j \in R} \sum_{l' \le l} \sum_{\rho \in \mathcal{DR}} Pr[x^*_{jil} = 1] \cdot Pr[s^*_{j,\rho} = 1] \cdot \min\{\rho, \frac{l \cdot C_l}{C_{unit}}\} \\
&= \sum_{r_j \in R} \sum_{l' \le l} Pr[x^*_{jil} = 1] \sum_{\rho \in \mathcal{DR}} Pr[s^*_{j,\rho} = 1] \cdot \min\{\rho, \frac{l \cdot C_l}{C_{unit}}\} \\
&= \sum\nolimits_{r_j \in R} \sum\nolimits_{l' \le l} x^*_{jil'} \cdot E(\min\{\rho, (l \cdot C_l)/C_{unit}\}). \qquad (22)
\end{aligned}$$

where $Pr[Y]$ is the probability that event Y is true. Constraint (10) is met by the optimal solution Opt.

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

$$E(Opt_j \mid x^*_{jil} = 1) = ER_{jil}. \qquad (23)$$

Considering all possible resource slots and base stations, we have

$$E(Opt_j) = \sum_{l,i} x^*_{jil} ER_{jil}. \qquad (24)$$

Therefore, $LPOpt \ge Opt$. ■

Proof of Lemma 2

Proof: Let Y be the amount of computing resource occupied by existing requests in the system. We need to show that

$$Pr[Y \ge l \cdot C_l] \le 1/2. \qquad (25)$$

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

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

where $s_{j',\rho}$ is a binary variable indicating whether request $r_{j'}$ has a data rate of ρ.

Taking expectation on both sides of Inequality (26), we have

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

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

$$\begin{aligned}
E(Y) &= \frac{1}{4} \sum_{j'} \sum_{l' \le l} \sum_{bs_i \in \mathcal{BS}} y_{jil'} \cdot E(\min\{\rho_{j'}, \frac{l \cdot C_l}{C_{unit}}\}) \qquad (26) \\
&\le (l \cdot C_l)/2, \text{due to constraint (9).} \qquad (27)
\end{aligned}$$

Applying Markov's inequality, we have

$$Pr[Y \ge l \cdot C_l] \le 1/2. \qquad (28)$$

This concludes the proof. ■

Proof of Theorem 1

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, a request may be rejected if existing requests occupy more than the amount $l \cdot C_l$ of computing resource of the base station. As shown in Lemma 2, such an event probability is smaller than $1/2$. On contrary, the probability of assigning request r_j to base station bs_i is greater than $1/2$. Let V_j be the random variable that corresponds to the amount of reward that algorithm `Appro` can deliver by admitting request r_j. We then have

$$E(V_j) \ge 1/8 \sum_{l} y_{jil} ER_{jil} = (1/8) LPOpt \ge (1/8) Opt,$$

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

Proof of Theorem 2

Proof: We first show the solution feasibility of algorithm `Heu`. Recall that the solution obtained by 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 assigning the tasks of each request to its nearby base stations. The re-assignment of a task of request r_j ensures its latency requirement and base station's capacity constraint to be met. Algorithm `Heu` thus delivers a feasible solution.

We then analyze the time complexity of algorithm `Heu`, which consists of the time for solving the **LP**, and the time for request pre-assignment.

It can be seen that **LP** has $O(|R| \cdot |\mathcal{BS}| \cdot L)$ decision variables. According to Cohen et al. [10], 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 derived.

The examination of each pre-assignment obtained can be conducted in $O(|R| \cdot |\mathcal{BS}|)$ time. 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 $n = |R| \cdot |\mathcal{BS}| \cdot L$ and the current values of ω and α are 2.37 and 0.31, respectively. ■

Proof of Theorem 3

Proof: We first define the expected total reward by algorithm 3. In algorithm 3, we discretize the continuous set of arms Z to Z' with κ arms. It can be seen 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* will incur, i.e.,

$$DE(Z') = ER^*(Z) - ER^*(Z'). \tag{29}$$

If a^* is the best arm in Z, and b is the closest arm in Z' to a^* in Z, we have $|a^* - b| \leq \epsilon$. By the Lipschitz condition in Eq. (12), we have

$$DE(Z') \leq \eta\epsilon. \tag{30}$$

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

$$\begin{aligned} E(R(T)) &= T \cdot ER^*(Z) - W(\texttt{DynamicRR}) \\ &= (T \cdot ER^*(Z') - W(\texttt{DynamicRR})) \\ &\quad + T \cdot (ER^*(Z) - ER^*(Z')) \\ &= R_S(T) + T \cdot DE(Z'), \end{aligned} \tag{31}$$

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

Following the study in [45], the regret bound of a successive elimination algorithm is $O(\sqrt{\kappa T \log T})$, which means that

$$R_S(T) = O(\sqrt{\kappa T \log T}). \tag{32}$$

By (32) and (30), the regret by algorithm DynamicRR is upper bounded by $E(R(T)) \leq O(\sqrt{\kappa T \log T} + T \cdot \eta \cdot \epsilon)$. ■

Proof of Theorem 4

Proof: The proof of the solution feasibility is similar to the one for algorithm DynamicRR, omitted.

We now analyze the time complexity of algorithm DynamicFair. For **Stage 1**, the time complexity of MT^2-LSTM prediction relies on the number of edges in a neural network [19]. Let W be the number of edges in its neural network. The time complexity of **Stage 1** for prediction thus is $O(W)$. In **Stage 2**, algorithm mainly performs segment partitioning, which has time complexity of $O(|\mathcal{BS}|)$. The most time-consuming part of **Stage 3** is the invoking of algorithm Heu iteratively. According to Theorem 2, each invocation of algorithm Heu takes $O(|R| \cdot |\mathcal{BS}| + (n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I \log^k((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I))$ time, where the current values of ω and α are 2.37 and 0.31, respectively. In the worse case, there are $|R|$ iterations. The time complexity of **Stage 3** thus is $O(|R|^2 \cdot |\mathcal{BS}| + |R|(n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I \log^k((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I))$. The time complexity of algorithm DynamicFair thus is $O(W + |R|^2 \cdot |\mathcal{BS}| + |R|(n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I \log^k((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})I))$. ■

REFERENCES

[1] N. Abbas, S. Sharafeddine, A. Mourad, C. Abou-Rjeily, and W. Fawaz, "Joint computing, communication and cost-aware task offloading in D2D-enabled Het-MEC," *Comput. Netw.*, vol. 209, May 2022, Art. no. 108900.

[2] S. K. Baruah, "Fairness in periodic real-time scheduling," in *Proc. 16th IEEE Real-Time Syst. Symp.*, 1995, pp. 200–209.

[3] X. Bei, Z. Li, and J. Luo, "Fair and efficient multi-resource allocation for cloud computing," in *Proc. WINE*, 2022, pp. 169–186.

[4] X. Bi, X. Sun, Z. Lyu, B. Zhang, and X. Wei, "A back adjustment based dependent task offloading scheduling algorithm with fairness constraints in VEC networks," *Comput. Netw.*, vol. 223, Mar. 2022, Art. no. 109552, doi: 10.1016/j.comnet.2022.109552.

[5] S. Bohez, J. D. Turck, T. Verbelen, P. Simoens, and B. Dhoedt, "Mobile, collaborative augmented reality using cloudlets," in *Proc. Int. Conf. MOBILe Wireless MiddleWARE, Operating Syst., Appl.*, Nov. 2013, pp. 45–54.

[6] T. Braud, P. Zhou, J. Kangasharju, and P. Hui, "Multipath computation offloading for mobile augmented reality," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2020, pp. 1–10.

[7] Z. Chen et al., "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–14.

[8] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10843–10856, Jul. 2021.

[9] J. Ren et al., "An efficient two-layer task offloading scheme for MEC system with multiple services providers," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 1519–1528.

[10] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," in *Proc. STOC*. New York, NY, USA: ACM, Jun. 2019, pp. 938–942.

[11] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, and J. Ott, "Nimbus: Towards latency-energy efficient task offloading for AR services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1530–1545, Apr./Jun. 2023, doi: 10.1109/TCC.2022.3146615.

[12] P. Cong, G. Xu, T. Wei, and K. Li, "A survey of profit optimization techniques for cloud providers," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–35, Mar. 2021.

[13] 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 Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1040–1055, May 2020.

[14] S. Duan et al., "MOTO: Mobility-aware online task offloading with adaptive load balancing in small-cell MEC," *IEEE Trans. Mobile Comput.*, early access, Nov. 8, 2022, doi: 10.1109/TMC.2022.3220720.

[15] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1414–1422.

[16] Z. Fang, X. Li, and R. Fan, "A mobile edge computing task offloading framework based on improved beetle antennae search," in *Proc. IEEE Int. Conf. Recent Adv. Syst. Sci. Eng. (RASSE)*, Dec. 2021, pp. 1–6.

[17] M. Gattullo, A. Evangelista, A. E. Uva, M. Fiorentino, and J. L. 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 Trans. Vis. Comput. Graphics*, vol. 28, no. 2, pp. 1443–1456, Feb. 2022.

[18] Z. Huang and V. Friderikos, "Proactive edge cloud optimization for mobile augmented reality applications," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2021, pp. 1–6.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[20] X. He et al., "History-assisted online user allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2022, pp. 140–149.

[21] S. Im, J. Kulkarni, and B. Moseley, "Temporal fairness of round robin: Competitive analysis for ℓ_k-norms of flow time," in *Proc. 27th ACM Symp. Parallelism Algorithms Architectures*, Jun. 2015, pp. 155–160.

[22] A. Jakl et al., "Enlightening patients with augmented reality," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Mar. 2020, pp. 195–203.

[23] 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," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2018, pp. 401–411.

[24] M. Jia and W. Liang, "Delay-sensitive multiplayer augmented reality game planning in mobile edge computing," in *Proc. 21st ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst.*, Oct. 2018, pp. 147–154.

[25] Z. Jing, Q. Yang, Y. Wu, M. Qin, K. Sup Kwak, and X. Wang, "Adaptive cooperative task offloading for energy-efficient small cell MEC networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022, pp. 292–297.

[26] I. Kash, A. D. Procaccia, and N. Shah, "No agent left behind: Dynamic fair division of multiple resources," *J. Artif. Intell. Res.*, vol. 51, pp. 579–603, Nov. 2014.
[27] M. Karimzadeh, S. M. Schwegler, Z. Zhao, T. Braun, and S. Sargento, "MTL-LSTM: Multi-task learning-based LSTM for urban traffic flow forecasting," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2021, pp. 564–569.
[28] H. Li, K. D. R. Assis, S. Yan, and D. Simeonidou, "DRL-based long-term resource planning for task offloading policies in multiserver edge computing networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4151–4164, Dec. 2022, doi: 10.1109/TNSM.2022.3191748.
[29] L. H. Lee et al., "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," 2020, *arXiv:2110.05352*.
[30] X. Li, L. Huang, H. Wang, S. Bi, and Y. A. Zhang, "An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 170–177, Feb. 2022.
[31] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2019, pp. 1–16.
[32] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2326–2335.
[33] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 1842–1850.
[34] L. Zhang, X. Wu, F. Wang, A. Sun, L. Cui, and J. Liu, "Edge-based video stream generation for multi-party mobile augmented reality," *IEEE Trans. Mobile Comput.*, early access, Dec. 27, 2022, doi: 10.1109/TMC.2022.3232543.
[35] Y. Liu, H. Xu, and W. C. Lau, "Online job scheduling with resource packing on a cluster of heterogeneous servers," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1441–1449.
[36] X. Li, X. Zhang, and T. Huang, "Asynchronous online service placement and task offloading for mobile edge computing," in *Proc. SECON*, 2022, pp. 1–9.
[37] C. Li, Y. Zhang, Q. Sun, and Y. Luo, "Collaborative caching strategy based on optimization of latency and energy consumption in MEC," *Knowl.-Based Syst.*, vol. 233, Dec. 2021, Art. no. 107523.
[38] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.
[39] Z. Ma et al., "Towards revenue-driven multi-user online task offloading in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1185–1198, May 2022.
[40] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2076–2085.
[41] G. S. Park, R. Kim, and H. Song, "Collaborative virtual 3D object modeling for mobile augmented reality streaming services over 5G networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3855–3869, Jul. 2023, doi: 10.1109/TMC.2022.3149543.
[42] X. Pang, Z. Wang, J. Li, R. Zhou, J. Ren, and Z. Li, "Towards online privacy-preserving computation offloading in mobile edge computing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 1179–1188.
[43] 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," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–9.
[44] X. Shang, Y. Huang, Y. Mao, Z. Liu, and Y. Yang, "Enabling QoE support for interactive applications over mobile edge with high user mobility," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 1289–1298.
[45] A. Slivkins, "Introduction to multi-armed bandits," 2019, *arXiv:1904.07272*.
[46] M. Song, Y. Lee, and K. Kim, "Reward-oriented task offloading under limited edge server power for multiaccess edge computing," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13425–13438, Sep. 2021.
[47] S. Shahsavari, F. Shirani, M. A. Khojastepour, and E. Erkip, "Opportunistic temporal fair mode selection and user scheduling in full-duplex systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1632–1651, May 2022.
[48] P. Si, J. Zhao, H. Han, K.-Y. Lam, and Y. Liu, "Resource allocation and resolution control in the metaverse with mobile augmented reality," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 3265–3271.
[49] J. Tan, W. Liu, T. Wang, M. Zhao, A. Liu, and S. Zhang, "A high-accurate content popularity prediction computational modeling for mobile edge computing using matrix completion technology," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, p. e3871, Jun. 2021.
[50] *Verizon*. Accessed: Nov. 2022. [Online]. Available: https://www.thefastmode.com/technology-solutions/15725-verizon-develops-new-5g-edge-technology-for-vr-mr-and-ar
[51] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Trans. Comput.*, vol. 71, no. 10, pp. 2449–2461, Oct. 2022.
[52] H. Wang, B. Kim, J. L. Xie, and Z. Han, "LEAF+AIO: Edge-assisted energy-aware object detection for mobile augmented reality," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5933–5948, Oct. 2023, doi: 10.1109/TMC.2022.3179943.
[53] H. Wang and J. Xie, "You can enjoy augmented reality while running around: An edge-based mobile AR system," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Dec. 2021, pp. 381–385.
[54] X. Wang, J. Ye, and J. C. S. Lui, "Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 1199–1208.
[55] Z. Xu et al., "Online learning algorithms for offloading augmented reality requests with uncertain demands in MECs," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 1064–1074.
[56] H. Xu, Y. Liu, and W. C. Lau, "Optimal job scheduling with resource packing for heterogeneous servers," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1553–1566, Aug. 2021.
[57] Z. Xiao et al., "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023, doi: 10.1109/TMC.2022.3199876.
[58] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2066–2075.
[59] Z. Xu et al., "Energy-aware inference offloading for DNN-driven applications in mobile edge clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 799–814, Apr. 2021.
[60] T. Yang, R. Chai, and L. Zhang, "Latency optimization-based joint task offloading and scheduling for multi-user MEC system," in *Proc. 29th Wireless Opt. Commun. Conf. (WOCC)*, May 2020, pp. 1–6.
[61] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading dependent tasks in mobile edge computing with service caching," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 1997–2006.

Zichuan Xu (Member, IEEE) received the B.Sc. and M.E. degrees in computer science from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the Ph.D. degree from The Australian National University in 2016. From 2016 to 2017, he was a Research Associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently a Full Professor with the School of Software, Dalian University of Technology. He is also a "Xinghai Scholar" with the Dalian University of Technology. His research interests include mobile edge computing, serverless computing, network function virtualization, and algorithm design.

Zhao Yuan received the B.E. degree in computer science and technology from the Wuhan University of Technology, China, in 2021. He is currently pursuing the M.E. degree in software engineering with the Dalian University of Technology. His research interests include mobile edge computing, augmented reality, and the Internet of Things.

Weifa Liang (Senior Member, IEEE) received the B.Sc. degree in computer science from Wuhan University, China, in 1984, the M.E. degree in computer science from the University of Science and Technology of China in 1989, and the Ph.D. degree in computer science from The Australian National University in 1998. He is currently a Full Professor with the Department of Computer Science, City University of Hong Kong. Prior to that, he was a Full Professor with The Australian National University. His research interests include design and analysis of energy efficient routing protocols for the Internet of Things and digital twins, mobile edge computing, network function virtualization, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He serves as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS.

Dongqi Liu received the B.E. degree in software engineering from the Dalian University of Technology, China, in 2020, where he is currently pursuing the M.E. degree in software engineering. His research interests include edge computing, task offloading, and resource allocation.

Wenzheng Xu (Member, IEEE) received the B.Sc., M.E., and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He was a Visitor with The Australian National University. He is currently an Associate Professor with Sichuan University. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Haipeng Dai (Senior Member, IEEE) received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2010, and the Ph.D. degree from Nanjing University in 2014. He is currently an Associate Professor with the Department of Computer Science and Technology, Nanjing University. His research interests include the Internet of Things, mobile computing, and data mining. His research papers appear in many prestigious conferences and journals, such as ACM MobiSys, ACM UbiComp, IEEE INFOCOM, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON MOBILE COMPUTING. He is a member of ACM. He received the Best Paper Award from IEEE ICNP'15, the Best Paper Award Runner-Up from IEEE SECON'18, and the Best Paper Award Candidate from IEEE INFOCOM'17. He serves/served as the Poster Chair for the IEEE ICNP'14 and the Track Chair for the ICCCN'19.

Qiufen Xia (Member, IEEE) received the B.Sc. and M.E. degrees in computer science from the Dalian University of Technology, China, in 2009 and 2012, respectively, and the Ph.D. degree in computer science from The Australian National University in 2017. She is currently an Associate Professor with the Dalian University of Technology. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.

Pan Zhou (Member, IEEE) received the B.S. degree in the advanced class from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006, the M.S. degree from the Department of Electronics and Information Engineering, HUST, in 2008, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in 2011. He was a Senior Technical Member with Oracle Inc., Boston, MA, USA, from 2011 to 2013. He is currently an Associate Professor with the School of Electronic Information and Communications, HUST. His research interests include big data analytics and machine learning, security and privacy, and information networks.