




Online Learning Aided Decentralized Multi-User Task Offloading for Mobile Edge Computing

Xiong Wang , Member, IEEE, Jiancheng Ye , Member, IEEE, and John C.S. Lui , Fellow, IEEE

Abstract—Mobile edge computing facilitates users to offload computation tasks to edge servers for meeting their stringent delay requirements. Previous works mainly explore task offloading when system-side information is given (e.g., server processing speed, cellular data rate), or centralized offloading under system uncertainty. But both generally fall short of handling task placement involving many coexisting users in an uncertain environment. In this paper, we develop a *multi-user* offloading framework considering *unknown yet stochastic* system-side information to enable a *decentralized user-initiated* service placement under overlapping server coverage. Specifically, we formulate the dynamic task placement as an online multi-user multi-armed bandit process, and propose a decentralized epoch based offloading (DEBO) to optimize user rewards which are subjected under network delay. We consider both cases without and with neighboring edge feedback once users' tasks are processed, where the latter incorporates system-side information sharing among edge servers for an enhanced task placement. For both cases, we show that DEBO can gradually deduce the optimal user-server assignment during dynamic offloading, thereby achieving a *close-to-optimal* service performance and *tight* $O(\log_2 T)$ regret. Moreover, we generalize DEBO to various common scenarios such as unknown reward gap, dynamic entering or leaving of clients, and fair reward distribution, while further exploring when users' offloaded tasks require *heterogeneous* computing resources. Particularly, we accomplish a *sub-linear* regret for each of these instances. Real measurements based evaluations corroborate the superiority of our offloading schemes over state-of-the-art approaches in optimizing delay-sensitive rewards.

Index Terms—Bandit feedback, mobile edge computing, multi-user offloading, uncertain system-side information.

Manuscript received 25 September 2022; revised 3 March 2023; accepted 1 May 2023. Date of publication 12 May 2023; date of current version 6 March 2024. The work was supported in part by Hubei Provincial Natural Science Foundation of China under Grant 2022CFB611, in part by NSFC under Grant 62202185, in part by the Research Grants Council (RGC) under Grant GRF 14215722, and in part by The Chinese University of Hong Kong (CUHK) under Grant 6905407. An earlier version of this paper was presented in the proceedings of IEEE INFOCOM 2022 [DOI: 10.1109/INFOCOM48880.2022.9796961]. Recommended for acceptance by W. Liao. (Corresponding author: Jiancheng Ye.)

Xiong Wang is with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab/Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xiongwang@hust.edu.cn).

Jiancheng Ye is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: jcy@connect.hku.hk).

John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMC.2023.3275851>, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3275851

I. INTRODUCTION

THE recent proliferation of smart devices has brought enormous popularity of many intelligent mobile applications (e.g., real-time face recognition, interactive gaming) which typically demand low latency and intensive computation [1]. Driven by emerging 5 G and IoT, 90% of the data will be generated and stored at the network edge [2], making it difficult for resource-constrained mobile devices to handle such huge amount of data. To address this challenge, mobile edge computing (MEC) has emerged as a new paradigm to push cloud frontier near to the network edge for supporting computation-intensive yet delay-sensitive applications [3].

With the availability of computing functionalities at the edge, MEC can facilitate mobile users to offload computation tasks to nearby edge servers, which are usually co-located with small-cell base stations or Wi-Fi access points. Under MEC, users need to determine the service placement of their offloaded tasks so as to shorten computing latency and enhance service performance. A typical MEC system is often divided into cell regions due to limited and overlapping radio coverage of edge servers as shown in Fig. 1, where users can be served by nonidentical servers when in their vicinity. This may lead to the performance disparity of edge computing caused by user roaming across different service regions. Therefore, one of the core problems is to make *effective offloading decision* to meet users' stringent delay requirements and augment the computing services [4].

Compared to the server-managed offloading scheme, *user-initiated* task placement enables a better personalized service support tailored to their individual preference, especially when edge servers are managed by different operators [5]. However, user mobility along with the stochastic MEC environment would give rise to a time-varying service performance. Worse yet, the system-side information (e.g., server processing speed, transmission data rate) is usually undisclosed to mobile users, which forces the user-initiated offloading to depend on previously perceived results.

There have been various efforts devoted to task offloading in MEC systems, so as to mitigate task delay for improving computing service. Nevertheless, they generally require the complete system-side information to aid the task placement design [4], [5], [6], which will be ineffective when this information is unknown or stochastically changing. Few works address system uncertainty via online learning based offloading schemes for service augmentation [7], [8], [9]. However, they mainly focus on *centralized offloading* with task placement decided by a *single*

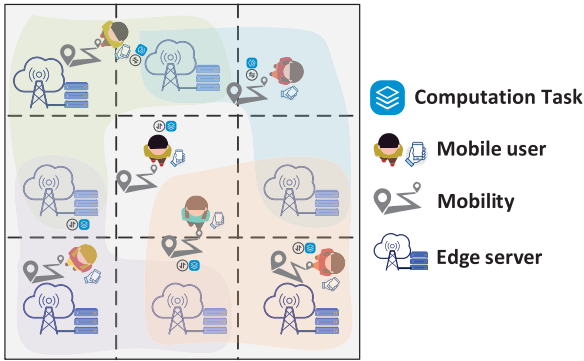


Fig. 1. MEC system with overlapping radio coverage.

user, while ignoring the *mutual influence* of coexisting users and the *capacity limitation* of edge servers. In practice, users in MEC systems need to share the edge resources to handle computation-intensive tasks, and at the same time, are *agnostic* of the system-side information and other users' demands. A critical question we want to address is “*how to characterize a decentralized offloading for many coexisting users in an uncertain and stochastic MEC environment*”. To answer this question, researchers are faced with the following challenges.

First, unknown system-side information demands a *learning* based adaptive offloading. In general, adaptive methods need to balance the exploration-exploitation trade-off, while the goal of achieving optimal performance for many coexisting users further complicates the offloading design. Second, a fully *decentralized* placement scheme *without inter-user communications* is desired as users are unaware of each other's existence when the MEC system scales. Worse still, only *noisy observations* can be perceived due to user mobility and the stochastic MEC environment, and hence the design of decentralized offloading scheme is forced to rely on bandit feedback. Though decentralized policies have been proposed [10], [11], [12], they are mainly built on the “collision-based” model, that is, all users would get zero rewards if making the same action, which are obviously inapplicable to model MEC systems. Third, users are covered by different servers' radio range and often have *various latency-sensitivity*, which inevitably lead to distinct offloading rewards. This requires balancing their delay-sensitive rewards to ensure a fair edge resource allocation with the consideration of server coverage overlapping. Fourth, edge servers are endowed with *limited computing capacity*, whereas different computation tasks could consume heterogeneous resource. The decentralized offloading ought to achieve a theoretically good performance while also respecting the capacity limit to avoid service blockage.

In this paper, we propose a fully decentralized multi-user offloading scheme for MEC which does not disclose the system-side information. Considering different user latency-sensitivity, we first devise a *preference function* to characterize the offloading reward subjected under task delay, which facilitates mitigating computing latency by leveraging the *perceived* reward feedback. On this basis, we formulate the dynamic task placement as an online *multi-user multi-armed bandit* (MAB)

process due to the uncertain MEC environment, where offloading to a reachable edge server is regarded as playing an arm. Since users can also handle tasks locally using handheld mobile devices whose information (processing speed) is given a priori, we regard all users' devices as a *virtual edge server* so as to unify the local task execution and offloading to servers at the same time. We develop a *decentralized* epoch based offloading (DEBO) scheme to balance the offloading exploration and exploitation. As a result, the *asymptotically optimal* rewards can be achieved by finally deriving the optimal user-server assignment in a decentralized manner, only using historically perceived observations. Besides, we advocate a heterogeneous DEBO (H-DEBO) to accommodate users' heterogeneous requirements. We show that H-DEBO can ensure a good service performance even when the oracle optimal assignment is unavailable. This paper makes the following contributions.

- We develop a *decentralized offloading framework* for a dynamic MEC system with overlapping radio coverage. Our scheme achieves close-to-optimal performance for coexisting users without any inter-user communications or the system-side information. To the best of our knowledge, this is the *first* work that conducts a thorough analysis of decentralized offloading under MEC system uncertainty.
- We propose DEBO to divide the time horizon into epochs for the dynamic task placement, where each epoch consists of an exploration, matching and exploitation phase to make the optimal placement decision, i.e., local processing or offloading to server. We investigate both cases without and with edge information sharing, in which the latter further incorporates neighboring edge cooperation. For both cases, DEBO attains a *tight* $O(\log_2 T)$ performance regret by merely using *bandit* reward feedback.
- We extend DEBO to general settings, i.e., unknown reward gap, dynamic user entering or leaving and fair reward distribution, and further quantify a *sub-linear regret* for each of these extensions. More importantly, we devise H-DEBO to handle the *heterogeneous* offloading requirements, where an $O(\log_2 T)$ regret is attained by solving an APX-hard assignment problem only with *learned* results.
- Extensive evaluations based on real-world measurements are performed to show the superiority of our offloading schemes over the existing approaches. Specifically, we can achieve *55.11% fairness improvement* by only *sacrificing 0.99% rewards* when incorporating the fair reward distribution into offloading design.

In the rest, we formulate the system model in Section II and illustrate DEBO in Section III. Next, Section IV extends DEBO to various settings and Section V presents H-DEBO. Section VI shows the performance evaluation and Section VII surveys related works. Finally, Section VIII concludes.

II. SYSTEM MODEL

We consider an MEC system with a set of mobile users $\mathcal{N} = \{1, 2, \dots, N\}$, and edge servers $\mathcal{K} = \{1, 2, \dots, K\}$ which can provide computing services within their radio range $\{C_j, j \in \mathcal{K}\}$. When the MEC system scales, users only retain local *user*

indices and need to make offloading decisions *independently* as they are agnostic of each other. Considering the partial coverage of edge servers, any user i could reach the servers $\mathcal{K}_i \subset \mathcal{K}$ if and only if being located in their radio range $\mathcal{C}_j, j \in \mathcal{K}_i$ as denoted by the shadow area in Fig. 1. Accordingly, we refer to \mathcal{K}_i as i 's reachable servers with $K_i = |\mathcal{K}_i|$, and \mathcal{N}_j as the covered users by server j with $N_j = |\mathcal{N}_j|$. Also, assume that $\mathcal{K}_i \cap \mathcal{K}_k \neq \emptyset, \exists i, k \in \mathcal{N}$, otherwise the MEC system is divided into disjoint subsystems.

A. User Task Offloading

At any time $t \in \{1, 2, \dots, T\}$, each user will offload a computation-intensive task to a *reachable server* or process the task locally by his mobile device, where the total time horizon T is *unspecified*. Suppose s_j and f_j are the potential transmission rate and processing speed of server j , respectively. Also, denote $\Theta = \{(s_j, f_j), j \in \mathcal{K}\}$ as the *system-side information*, which is often *undisclosed* to users [8], [9]. Regarding user i , the processing speed of its device is l_i , which however is known to user i by directly reading the CPU information. For consistency purposes, we collectively refer to all users' devices as the *virtual edge server* K' , so that any user i offloads or processes a task could be unified by offloading the task to a server $j \in \mathcal{K}_i \cup \{K'\}$.

An offloaded task from user i is typically characterized by the task size b_i (e.g., amount of cellular traffic) and required CPU cycles per unit traffic γ_i [13]. If i 's task is handled by server $j \in \mathcal{K}_i$, it will experience an offloading delay d_{ij} , including the transmission and processing time:

$$d_{ij} = \frac{b_i}{s_j} + \frac{b_i \gamma_i}{f_j}. \quad (1)$$

Similarly, when the task is executed by virtual server K' (processed locally), the delay $d_{i,K'}$ signifies the processing time as there is no transmission involved:

$$d_{i,K'} = \frac{b_i \gamma_i}{l_i}. \quad (2)$$

Along with the offloading delay $d_{ij}, \forall j \in \mathcal{K}_i \cup \{K'\}$, user i also associates an instant reward μ_{ij} to represent its personal preference on the service performance:

$$\mu_{ij} = v_i - g_i(d_{ij}), \quad (3)$$

where v_i is the intrinsic task value and $g_i(\cdot)$ is a cost function which increases with the delay, indicating the latency-sensitivity of user i . We are interested in the non-trivial case where $v_i > g_i(d_{ij})$, i.e., users acquire positive rewards after successful task offloading. Due to unknown information Θ and system uncertainty (e.g., user mobility, transmission/processing oscillation), each user can only perceive an i.i.d. random reward value at time t , with $\mu_{ij} = \mathbb{E}[r_{ij}(t)], r_{ij}(t) \in [\underline{r}, \bar{r}], \forall i \in \mathcal{N}, j \in \mathcal{K}_i$ where positive \underline{r} and \bar{r} are the lower and upper reward bounds. In contrast, any user i could acquire the exact service reward $\mu_{i,K'} \in [\underline{r}, \bar{r}]$ since its local processing speed l_i is acknowledged beforehand.

In general, the knowledge difference between the rewards of server offloading and local processing requires a distinct offloading decision making. Specifically, the reward $\mu_{ij}, j \in \mathcal{K}_i$,

i.e., offloading to "real" server, is to be explored using observations $r_{ij}(t)$ while $\mu_{i,K'}$, offloading to virtual server, is given in advance. Besides, we will further investigate the case of neighboring edge feedback, where edge servers would *share* their current oscillated processing speed and transmission rate with each other to enable an enhanced user task placement. Hereinafter, we will leverage the prefix "virtual" to distinguish "real" servers from mobile devices, and also use virtual server and mobile device interchangeably in line with the context.

B. Server Computing Capacity

Compared to the cloud datacenter, an edge server essentially has *limited computing capacity* [14], which we rephrase as maximum task service or endowed computing resource depending on the user offloading requirement.

1) *Maximum Task Service*: When users need homogeneous computing resource (memory, CPU, storage) to process their offloaded tasks, the task service capacity M_j of server $j \in \mathcal{K}$ represents how many tasks it can handle concurrently. If excessive computation tasks arrive, the server will randomly choose M_j tasks while abandoning the rest due to its limited capacity. Once discarded, the user experiences a high delay and observes a zero reward $r_{ij}(t) = 0$. Besides, denote $M = \sum_{j=1}^K M_j$ as the total MEC service capacity of all edge servers.

2) *Endowed Computing Resource*: We also consider the situation where users may require heterogeneous resource for processing their offloaded tasks, say running different types of applications. For this case, the capacity C_j of any edge server $j \in \mathcal{K}$ stands for the amount of its endowed computing resource.

C. Problem Formulation

1) *Homogeneous Resource Requirement*: Under the homogeneous requirement, computing capacity of each server implies the maximum number of admitted tasks. At time t , denote $a_i(t) \in \mathcal{K}_i \cup \{K'\}$ as the selected (virtual) server decision of user i with $\mathbf{a}(t) = \{a_i(t), i \in \mathcal{N}\}$, and $\mathcal{N}_j(t) = \{i, a_i(t) = j\}$ as the user set choosing server j . Users aim to mitigate the computing delay while avoiding task rejection due to violating the server capacity. Let \mathbf{a}^* be the optimal offloading decisions made by an omniscient oracle when the *system-side information* Θ is known, which are the solution to the offline assignment problem (OAP) below:

$$\begin{aligned} \max \quad & \sum_{i=1}^N \mu_{ia_i} \\ \text{s.t.} \quad & |\mathcal{N}_j| \leq M_j, \forall j \in \mathcal{K}. \end{aligned} \quad (4)$$

Since mobile users are agnostic of each other, they can only decide $\mathbf{a}(t)$ in a *decentralized* fashion. Accordingly, we quantify the offloading performance by its regret, defined as the difference between accumulated rewards of $\mathbf{a}(t)$ and that of the oracle decisions \mathbf{a}^* to OAP:

$$\mathcal{R}(T) = T \sum_{i=1}^N \mu_{ia_i^*} - \sum_{t=1}^T \sum_{i=1}^N \mathbb{E}[r_{ia_i(t)}(t)]. \quad (5)$$

Note that $\mathbb{E}[r_{ia_i(t)}(t)] = \mu_{ia_i(t)}$ if offloaded task is processed remotely or locally, otherwise $r_{ia_i(t)}(t) = 0$ once being discarded. For ease of exposition, we also refer to \mathbf{a}^* or $\mathbf{a}(t)$ as an *assignment* between users \mathcal{N} and servers $\mathcal{K} \cup \{K'\}$.

2) *Heterogeneous Resource Requirement*: Consider each user i requires a unique c_i computing resource to handle his offloaded task. Let $C'_j(t) = \sum_{i \in \mathcal{N}_j(t)} c_i$ be the total resource request for server $j \in \mathcal{K}$ at time t . Again, the optimal offloading decision is marked as \mathbf{a}^* , which can be obtained by solving the heterogeneous offline assignment problem (H-OAP) given the system-side information Θ :

$$\begin{aligned} \max \quad & \sum_{i=1}^N \mu_{ia_i} \\ \text{s.t.} \quad & C'_j(t) \leq C_j, \forall j \in \mathcal{K}. \end{aligned} \quad (6)$$

Since the virtual server K' represents local task execution, it has abundant capacity to handle all offloading requests. Different from OAP, H-OAP is a typical generalized assignment problem (GAP) with no optimal solution for polynomial-time algorithms [15]. Moreover, any centralized sub-optimal solution \mathbf{a} to GAP only ensures at most $(1 + \alpha)$ -approximate compound reward, that is $\sum_{i=1}^N \mu_{ia_i} \geq \frac{1}{1+\alpha} \sum_{i=1}^N \mu_{ia_i^*}$, $\alpha \geq 1$. As a consequence, the offloading regret for the heterogeneous requirement is defined by leveraging the *highest guaranteed* sub-optimal reward $\frac{1}{2} \sum_{i=1}^N \mu_{ia_i^*}$, i.e., $\alpha = 1$:

$$\tilde{\mathcal{R}}(T) = \frac{T}{2} \sum_{i=1}^N \mu_{ia_i^*} - \sum_{t=1}^T \sum_{i=1}^N \mathbb{E}[r_{ia_i(t)}(t)]. \quad (7)$$

If the task is rejected due to any server capacity limitation, we have $r_{ia_i(t)}(t) = 0$. Also, offloading decisions or assignment $\mathbf{a}(t)$ should be determined by decentralized methods.

3) *Model Discussion*: Our main objective is to optimize the offloading performance through dynamic task placement in a decentralized fashion. This can only be achieved by *online learning* based methods due to the undisclosed system-side information and agnostic user coexistence. Another challenge is that because edge servers can accommodate multiple tasks and meanwhile users could merely send tasks to reachable servers, previous collision-based indirect collaboration frameworks are not applicable to our offloading design [10], [16]. Furthermore, the available information difference between edge servers and mobile devices calls for different reward explorations when devising the task offloading schemes. Last but not least, APX-hard H-OAP has no optimal solution in polynomial time caused by heterogeneous offloading requests, which in fact demands a distinct decentralized solution from the homogeneous OAP.

III. DECENTRALIZED TASK OFFLOADING SCHEME

In this section, we first discuss the design of offloading scheme for the homogeneous request, with a server capacity denoting the number of admitted tasks. In particular, the user-initiated task placement is modeled as a multi-user MAB problem by regarding offloading to an edge server as playing an arm, while it needs to be tackled via decentralized techniques.

Algorithm 1: DEBO: Decentralized Epoch Based Offloading.

Input: $\{M_j, j \in \mathcal{K}\}, M, T_1, T_2, \epsilon$
1: Initialization: Set $\mathbf{V} = \{V_{ij} = 0, \forall i \in \mathcal{N}, j \in \mathcal{K}_i\}$ and $\mathbf{S} = \{S_{ij} = 0, \forall i \in \mathcal{N}, j \in \mathcal{K}_i\}$;
2: **for** epoch $n = 1$ to n_T **do**
3: **Exploration:** $\tilde{\mathbf{r}}^{(n)} = \text{RO}(\mathbf{V}, \mathbf{S}, T_1)$;
4: **Matching:** $\mathbf{a}' = \text{DAuction}(\tilde{\mathbf{r}}^{(n)}, \boldsymbol{\mu}_{K'}, T_2, \epsilon)$;
5: **Exploitation:** for remaining 2^n time slots:
6: User i offloads tasks to (virtual) edge server a'_i ;

A. Epoch Based Time Division

To achieve satisfactory service performance or minimize the offloading regret $\mathcal{R}(T)$, we have to consider the exploration-exploitation trade-off for offloading to each server. Since the total time T is unspecified a priori, we divide the time horizon into epochs $\{1, 2, \dots, n_T\}$, where each epoch has a variable number of time slots and n_T is the last epoch index. To strike a balance between offloading exploration and exploitation, every epoch is composed of an exploration phase, matching phase and exploitation phase.

- **Exploration phase:** this phase lasts for T_1 time slots in each epoch. Users will randomly offload tasks to edge servers \mathcal{K} for acquiring the estimated rewards $\tilde{\mathbf{r}}^{(n)} = \{\tilde{r}_{ij}^{(n)}, i \in \mathcal{N}, j \in \mathcal{K}_i\}$. Concretely, each $\tilde{r}_{ij}^{(n)}$ is calculated by using *all explored observations* from the beginning to the current epoch n .
- **Matching phase:** this phase has a length of T_2 time slots. Users leverage a *decentralized auction*, which will be elaborated later, based on estimated rewards $\tilde{\mathbf{r}}^{(n)}$ and $\boldsymbol{\mu}_{K'} = \{\mu_{iK'}, i \in \mathcal{N}\}$ to yield an assignment \mathbf{a}' .
- **Exploitation phase:** this phase occupies 2^n time slots in epoch n . All users offload tasks obeying the assignment \mathbf{a}' to fully exploit the corresponding rewards.

The fact that the exploitation takes an exponential number of slots does not imply it occupies a long duration in practice, rather, it means the exploitation phase needs to *dominate* both the exploration and matching phases. In general, our epoch division ensures the convergence of \mathbf{a}' to the optimal assignment \mathbf{a}^* given an accurate reward estimation $\tilde{\mathbf{r}}^{(n)}$. We underscore that local processing rewards are involved in all three phases except for the exploration phase as each $\mu_{iK'}$ in (3) can be directly computed. Later analysis points out that this preclusion will shorten the exploration length T_1 .

B. Decentralized Epoch Based Offloading

1) *Algorithm Design*: We now formally specify the procedure of the decentralized epoch based offloading (DEBO) in Algorithm 1. In epoch n , each user *sequentially* performs the random offloading (RO) for the first T_1 time slots, then the decentralized auction (DAuction) for the next T_2 time slots, then the offloading exploitation for remaining 2^n time slots.

2) *Exploration of RO*: Users will locally execute the RO as presented in Algorithm 2. Since server $j \in \mathcal{K}$ can support M_j tasks, so we consider that it has M_j resource units and there are

Algorithm 2: RO: Random Offloading.

Input: $\{M_j, j \in \mathcal{K}\}, M, \mathbf{V}, \mathbf{S}, T_1$

- 1: **for** T_1 time slots **do**
- 2: Set $\mathcal{N}_j(t) = \emptyset, \forall j \in \mathcal{K}$;
- 3: **for** user $i \in \mathcal{N}$ **do**
- 4: Randomly choose an integer H_i in \mathcal{L}_i ;
- 5: Offload a task to the server $a_i = j$ if $H_i \in (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$; \triangleright Also send integer H_i
- 6: $\mathcal{N}_{a_i}(t) = \mathcal{N}_{a_i}(t) \cup \{i\}$;
- 7: **if** $r_{ij}(t) > 0$ **then**
- 8: $V_{ij} = V_{ij} + 1, S_{ij} = S_{ij} + r_{ij}(t)$;
- 9: **for** edge server $j \in \mathcal{K}$ **do**
- 10: **if** $|\mathcal{N}_j(t)| \leq M_j$ **then**
- 11: Process all offloaded tasks;
- 12: **else**
- 13: Randomly choose one task from users having the same integer $\{i, H_i = H\}$, and abandon the rest;
- 14: **return** $\tilde{\mathbf{r}}^{(n)} = \frac{\mathbf{S}}{\mathbf{V}}$; \triangleright Element-wise division

totally K servers containing M units where the index ranges $(\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$ represent the M_j units in server j . For user i , let $\mathcal{L}_i = \cup_{j \in \mathcal{K}_i} (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l] \subset [1, M]$ denote the resource units of its reachable servers \mathcal{K}_i . Accordingly, integer H_i in Line 4 entails the selected server index (resource unit) for task offloading, and enables task abandonment if received tasks exceed the server capacity in each slot (Line 13). Note that \mathbf{V} and \mathbf{S} in Line 8 record the *successful* offloading times (perceiving positive rewards) and accumulated rewards, respectively, which leverage all observations during the exploration to learn $\tilde{\mathbf{r}}^{(n)}$. Intuitively, the computational complexity and required extra message are both $O(1)$ for each user in each slot (Lines 4-5).

3) *Matching of DAuction:* The DAuction is shown in Algorithm 3. Specifically, DAuction uses the estimated rewards $\tilde{\mathbf{r}}^{(n)}$ and local processing rewards $\mu_{K'}$ to decide the assigned (virtual) edge server to each user, while the optimal assignment \mathbf{a}^* will be deduced if every learned reward $\tilde{r}_{ij}^{(n)}$ converges to the expected value μ_{ij} . In addition to the M resource units, L' represents the resource index of the virtual server.

Denote range $M(m) = (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$ and server index $j(m) = j$, if $m \in (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$, $\forall j \in \mathcal{K}$. As for the virtual server, i.e., $m = L'$, let $M(m) = L'$ and $j(m) = K'$. Lines 2-5 state the initialization of rewards \mathbf{R} for each user pertaining to $(|\mathcal{L}_i| + 1)$ resource units. To simplify notations, assignment $a_i \in \mathcal{L}_i \cup \{L'\}$ means that user i will offload tasks to server $j(a_i)$ by holding the resource unit a_i (Lines 12 and 14), while $a_i = 0$ (Line 8) implies the user *remains or returns* unassigned to any edge server. Based on whether we observe a positive reward (e.g., task is processed), a_i is updated accordingly. To derive the bid (Line 14), any user i will find the resource unit m^* with the highest value in Line 9 to offload a task, and further attain the resource m' with the second highest value in Line 10 by precluding those units belonging to the same server to expedite the auction process [19]. On servers' side, they will allocate their resource units (i.e., handling the

Algorithm 3: DAuction: Decentralized Auction.

Input: $\{M_j, j \in \mathcal{K}\}, M, \tilde{\mathbf{r}}^{(n)}, \mu_{K'}, T_2, \epsilon$

- 1: Initialization: $\mathbf{R} = \{R_{im}, \forall i \in \mathcal{N}, m \in \mathcal{L}_i \cup \{L'\}\}$, set $\mathbf{B} = \{B_{im} = 0, \forall i \in \mathcal{N}, m \in \mathcal{L}_i \cup \{L'\}\}$ and $\mathbf{a} = \mathbf{0}$;
- 2: **for** $i \in \mathcal{N}$ **do**
- 3: **for** $m \in \mathcal{L}_i$ **do**
- 4: $R_{im} = \tilde{r}_{ij}^{(n)}$ if $m \in (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$;
- 5: $R_{iL'} = \mu_{iK'}$ \triangleright Virtual edge server
- 6: **for** T_2 time slots **do**
- 7: **for** user $i \in \mathcal{N}$ **do**
- 8: **if** $a_i = 0$ **then**
- 9: Find $m^* \in \arg \max_m \{R_{im} - B_{im}\}$;
- 10: Find $m' \in \arg \max_{m \neq M(m^*)} \{R_{im} - B_{im}\}$;
- 11: **if** $m^* = L'$ **then**
- 12: Locally process the task and set $a_i = L'$;
- 13: **continue**;
- 14: Offload to server $j(m^*)$, bid $B_{im^*} = R_{im^*} - (R_{im'} - B_{im'}) + \epsilon$, observe reward, set $a_i = m^*$ if task is processed and $a_i = 0$ if abandoned;
- 15: **for** edge server $j \in \mathcal{K}$ **do**
- 16: If the $m \in (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^j M_l]$ th unit is allocated to user i , compare all received bids with B_{im} to select the highest-bid user and allocate m th unit;
- 17: If the m th unit is not allocated, select the highest-bid user to allocate the m th unit;
- 18: **for** $i \in \mathcal{N}$ **do**
- 19: $a'_i = j(a_i)$;
- 20: **return** $\mathbf{a}' = \{a'_i, i \in \mathcal{N}\}$;

offloaded tasks) to users who have the highest bids (Lines 15-17), and discard tasks from unassigned users. Note that the auction is decentralized as any user i only needs to locally maintain a reward vector $\{R_{im}, m \in \mathcal{L}_i \cup \{L'\}\}$, a bidding vector $\{B_{im}, m \in \mathcal{L}_i \cup \{L'\}\}$, and server indices a_i, a'_i . In any time slot, DAuction needs $O(M)$ computation to select the highest-value resource unit and $O(1)$ additional message to send the bid pertaining to each user.

C. Offloading With Edge Information Sharing

In practice, edge servers are often possessed by a private owner, like a content delivery network provider, or managed by different operators [20]. Occasionally, the owner can orchestrate its edge servers to share their information, like current processing speed or transmission rate, or various operators would exchange such information with each other. This shared edge information will help mobile users make wiser offloading decisions, which also benefits the edge servers in promoting their computing service.

Specifically, when user i *successfully* offloads a task to server j (task is processed), i not only attains a reward observation $r_{ij}(t)$, but also obtains the oscillated information of other nearby servers $\mathcal{K}_i \setminus \{j\}$ to virtually compute the noisy rewards $r_{ij'}(t), j' \in \mathcal{K}_i \setminus \{j\}$ due to neighboring edge feedback [21]. Actually, this can be achieved since server j is able to piggyback

the exchanged edge information when sending the processed task results to user i . Therefore, as an enhancement, we further explore the decentralized task placement and propose DEBO with edge information sharing. Similarly, time horizon is still divided into epochs $\{1, 2, \dots, n_T\}$, where each epoch also consists of the exploration, matching and exploitation phases. In particular, the exploration phase will use the observed and computed rewards to acquire the reward estimations $\tilde{r}^{(n)}$, which are the basis for deriving the optimal user-server matching and later reward exploitations. In broad strokes, the main difference when we have neighboring edge feedback lies in the random offloading. To be more specific, Lines 7-8 of RO in Algorithm 2 are changed to record the offloading times and accumulated rewards for all reachable servers \mathcal{K}_i , rather than only for server j when a user's task is processed. By doing so, we can accelerate the exploration phase, i.e., set a shorter T_1 . As for DAuction in the matching phase and the exploitation phase, they will remain unchanged. Next, we characterize DEBO without and with edge information sharing as a whole.

D. Performance Analysis of DEBO

1) *Main Regret Result*: We now analyze the regret $\mathcal{R}(T)$ in (5). Let $\Delta_{\min} = \min_{\mathbf{a} \neq \mathbf{a}^*} \{\sum_{i=1}^N \mu_{ia_i^*} - \sum_{i=1}^N \mu_{ia_i}\}$, which is the compound reward gap between the optimal and the best sub-optimal assignments. W.l.o.g., assume there is a unique optimal assignment \mathbf{a}^* to OAP, otherwise Δ_{\min} implies the gap between the highest and second highest compound rewards accordingly. Moreover, denote $M_{\min} = \min_{j \in \mathcal{K}} \{M_j\}$, $L_{\max} = \max_{i \in \mathcal{N}} \{|\mathcal{L}_i|\}$, $L_{\min} = \min_{i \in \mathcal{N}} \{|\mathcal{L}_i|\}$, $K_{\max} = \max_{i \in \mathcal{N}} \{K_i\}$, $N_{\max} = \max_{j \in \mathcal{K}} \{N_j\}$, and $v = \frac{N_{\max}}{L_{\min} - 1}$. Let $\delta_{\min} = \min_{i \in \mathcal{N}} \min_{j, j' \in \mathcal{K}_i \cup \{K'\}, j \neq j'} \{|\mu_{ij} - \mu_{ij'}|\}$ be the minimum user reward gap with both $\Delta_{\min}, \delta_{\min} > 0$. In fact, L_{\max} and N_{\max} are the potentially maximum server resource a user may access and the largest population an edge server will cover, respectively, and so do L_{\min}, K_{\max} .

Theorem 1: Let $\epsilon = \max\{\frac{\Delta_{\min}}{5N}, \frac{\delta_{\min}}{K+1} - \frac{3\Delta_{\min}}{4N(K+1)}\}$, $T_1 = \max\left\{\left\lceil \frac{32}{9} \frac{4^v N^2 L_{\max} (\bar{r}-r)^2}{\Delta_{\min}^2 M_{\min}} \right\rceil, \left\lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \right\rceil\right\}$ or $T_1 = \max\left\{\left\lceil \frac{32}{9} \frac{4^v N^2 (\bar{r}-r)^2}{\Delta_{\min}^2} \right\rceil, \left\lceil \frac{81}{32v} 16^v \right\rceil\right\}$ with edge information sharing, and $T_2 = \lceil N(L_{\max} + 1) + \frac{N L_{\max} \bar{r}}{\epsilon} \rceil$. The regret $\mathcal{R}(T)$ of DEBO is upper bounded by

$$\begin{aligned} \mathcal{R}(T) &\leq (T_1 N \bar{r} + T_2 N \bar{r}) \log_2(T + 2) + 12 N^2 K_{\max} \bar{r} \\ &= O(\log_2 T). \end{aligned} \quad (8)$$

See Appendix A.2, available online for the proof. The regret in (8) is obtained by *bounding the error probability* P_n that $\mathbf{a}' \neq \mathbf{a}^*$ after the n th matching, which is shown in Lemma 4. Since T_1 in the edge information sharing case is smaller, (8) also implies a theoretically tighter bound, or smaller coefficient of $O(\log_2 T)$, when we have information exchange.

Remark: If Δ_{\min} is too small, leading to a prohibitively long exploration length T_1 , one can adjust it to a sufficiently large value in practice since T_1 in Theorem 1 simply provides a loose upper bound. Moreover, the computational complexity

of DEBO is $O((T_1 + MT_2) \log_2 T)$ if combining with previous computation analysis for RO and DAuction.

In the following, we further demonstrate that our derived $\log_2 T$ regret is *tight* by regrading a user as a super-user who can control all users' offloading decisions [22].

Proposition 1: The regret $\mathcal{R}(T)$ defined in (5) of any offloading scheme is at least $\Omega(\log_2 T)$.

Please refer to Appendix A.3, available online for the detailed proof.

2) *Exploration Error Probability*: The goal of exploration RO is to estimate rewards accurately for the use in the matching phase, where the exploration error probability is presented in the next lemma with its proof in Appendix B, available online.

Lemma 1: Let $T_1 = \max\left\{\left\lceil \frac{32}{9} \frac{4^v N^2 L_{\max} (\bar{r}-r)^2}{\Delta_{\min}^2 M_{\min}} \right\rceil, \left\lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \right\rceil\right\}$ or $T_1 = \max\left\{\left\lceil \frac{32}{9} \frac{4^v N^2 (\bar{r}-r)^2}{\Delta_{\min}^2} \right\rceil, \left\lceil \frac{81}{32v} 16^v \right\rceil\right\}$ with edge information sharing. After the n th exploration, the error probability satisfies

$$\Pr\left(|\tilde{r}_{ij}^{(n)} - \mu_{ij}| > \frac{3\Delta_{\min}}{8N}\right) \leq 3NK_{\max} e^{-n}, \forall i \in \mathcal{N}, j \in \mathcal{K}_i. \quad (9)$$

Lemma 1 states that each estimated reward $\tilde{r}_{ij}^{(n)}$ is sufficiently close to expected reward μ_{ij} with high probability.

3) *Decentralized Matching Error*: Based on estimated and local processing rewards $\mu_{K'}$, users execute the DAuction to yield the user-server assignment \mathbf{a}' . If the compound reward induced by \mathbf{a}' is within a gap of Δ_{\min} to the highest outcome, then \mathbf{a}' is in fact the optimal assignment, or $\mathbf{a}' = \mathbf{a}^*$. Note that DAuction runs in a decentralized manner with each server accommodating many tasks concurrently, which distinguishes it from existing auctions requiring inter-user communications [17], [19] or focusing on one-to-one match [11], [23]. Before characterizing the matching error, we first show that the assignment in DAuction fulfills the ϵ -complementary slackness (ϵ -CS).

Lemma 2: Denote $\eta_m = \max_{i \in \mathcal{N}_j} \{B_{im}\}$ as the highest bid among the bidding users, and $\tilde{\eta}_{j(m)} = \min_{m \in M(m)} \{\eta_m\}$ as the price of edge server $j(m)$ in DAuction, then the resource unit assignment \mathbf{a} satisfies ϵ -CS, that is

$$R_{ia_i} - \tilde{\eta}_{j(a_i)} \geq \max_m \{R_{im} - \tilde{\eta}_{j(m)}\} - \epsilon. \quad (10)$$

See Appendix C.1, available online for the proof. ϵ -CS implies that the assignment \mathbf{a}' attains at least near to the optimal reward.

Lemma 3: With a slight abuse of notation, let $\tilde{r}_{iK'}^{(n)} = \mu_{iK'}, \forall i \in \mathcal{N}$ in this lemma. Denote $\mathbf{a}^{(n)}$ as the optimal assignment under $\tilde{\mathbf{r}}^{(n)}$, and $\tilde{\Delta}_{\min} = \min_{i \in \mathcal{N}, j, j' \in \mathcal{K}_i \cup \{K'\}, j \neq j'} \{|\tilde{r}_{ij}^{(n)} - \tilde{r}_{ij'}^{(n)}|\}$. If $(K+1)\epsilon < \tilde{\Delta}_{\min}$, DAuction ensures $\mathbf{a}' = \mathbf{a}^{(n)}$. Also:

$$\sum_{i=1}^N \tilde{r}_{ia_i}^{(n)} \geq \sum_{i=1}^N \tilde{r}_{ia_i}^{(n)} - N\epsilon. \quad (11)$$

Besides, DAuction will terminate with all users being assigned to servers within $T_2 = \lceil N(L_{\max} + 1) + \frac{N L_{\max} \bar{r}}{\epsilon} \rceil$ rounds.

See Appendix C.2, available online for the proof. In addition to the common $N\epsilon$ gap [11], [17], [23], our proposed DAuction further guarantees a $(K+1)\epsilon$ matching error. By using this,

we can facilitate a more precise calibration of the minimum increment ϵ in Theorem 1. This is because the server number K is usually much smaller than the user number N , so one can speed up DAuction by setting a larger ϵ .

4) *Error Probability of DEBO*: In line with Lemmas 1 and 3, we present the error probability P_n that $\mathbf{a}' \neq \mathbf{a}^*$ with the proof in Appendix C.3, available online.

Lemma 4: If setting the parameters as in Theorem 1, the error probability P_n that $\mathbf{a}' \neq \mathbf{a}^*$ is bounded $P_n \leq 3NK_{\max}e^{-n}$.

This lemma ensures that \mathbf{a}' obtained from DAuction will be optimal with increasingly high probability over epoch n .

IV. EXTENSION OF DECENTRALIZED OFFLOADING

The regret in Theorem 1 is materialized only when leveraging the reward gaps Δ_{\min} , δ_{\min} and constant user number N . This section extends previous regret analysis to show the robustness of our offloading scheme in more general settings, i.e., unknown reward gaps, dynamic user entering or leaving, and fair reward distribution.

A. Unknown Reward Gap

The knowledge of reward gaps Δ_{\min} , δ_{\min} may be unavailable in practice, which requires us to set the exploration or matching length adaptively. Remember that the exploration of RO aims to acquire an accurate reward estimation, and the matching of DAuction is to deduce the optimal user-server assignment. Therefore, one can still harness DEBO in Algorithm 1 by prolonging exploration length $T_1^{(n)}$ over epoch to ensure a bounded exploration error probability, meanwhile reducing the minimum increment $\epsilon^{(n)}$, or extending matching length $T_2^{(n)}$, so as to converge to the optimal assignment. Accordingly, we denote this decentralized offloading under unknown reward gaps as U-DEBO.

Theorem 2: Let $\epsilon^{(n)} = c_0n^{-\vartheta}$, $T_1^{(n)} = \lceil c_1n^\vartheta \rceil$ with- or without edge information sharing, and $T_2^{(n)} = \lceil N(L_{\max} + 1) + \frac{NL_{\max}\bar{r}}{\epsilon^{(n)}} \rceil$ where c_0 , c_1 are constants and $\vartheta \in (0, 1)$. The regret $\mathcal{R}(T)$ of U-DEBO is bounded by

$$\begin{aligned} \mathcal{R}(T) &\leq \lceil c_1 \log_2^\vartheta(T+2) \rceil N\bar{r} \log_2(T+2) \\ &+ \lceil N(L_{\max} + 1) + \log_2^\vartheta(T+2)NL_{\max}\bar{r}/c_0 \rceil \\ &\times N\bar{r} \log_2(T+2) + N\bar{r}(2^{n_0} - 1) + 12N^2K_{\max} = O(\log_2^{1+\vartheta} T), \end{aligned} \quad (12)$$

where n_0 is a finite integer.

Please refer to Appendix D.1, available online for the proof. Note that there is a power $(1 + \vartheta)$ on $\log T$, which stems from cautiously elongating $T_1^{(n)}$ and $T_2^{(n)}$ to ensure a bounded error probability analogous to that in Lemma 4.

B. Dynamic User Mobility

Mobile users may enter or leave the MEC system dynamically, leading to a time-varying user population N . When this happens, the user will notify its nearby edge server, so that the remaining users could perceive the dynamic value of N from

servers without inter-user communications. Sharing a similar spirit to [12], we postulate that signaling the entering or leaving occurs at the *beginning* of each epoch.

Denote $N^{(n)}$ as the user number in epoch n . Note that the user entering and leaving will have different impacts. For the leaving case, even all users adhere to DEBO with unchanged parameters T_1 and T_2 as in Theorem 1, it still yields an $O(\log_2 T)$ regret. This is because the error probability satisfies $P_n \leq 3N^{(n)}K_{\max}e^{-n}$ when running the RO and DAuction for longer time due to the reduction of user number $N^{(n)}$. Nevertheless, adjusting T_1 and T_2 along with $N^{(n)}$ can in fact reduce the empirical regret. In contrast, the dynamic entering would cause an increase in the population $N^{(n)}$, which brings about an unbounded error probability P_n as newly joined users have not yet experienced adequate explorations to acquire an accurate reward estimation for attaining the optimal assignment in the matching phase. It should be emphasized that we will observe a dynamically changing optimal assignment \mathbf{a}^* in either leaving or entering case. Similarly, refer to the decentralized offloading for dynamic user mobility as D-DEBO, whose regret compared to varying optimal assignments is quantified below.

Theorem 3: Suppose that the epoch n' of the last user entering satisfies $n' \leq O(\log_2 T^\zeta)$, $\zeta \in (0, 1)$. Let $\epsilon = \max\{\frac{\Delta_{\min}}{5N^{(n)}}, \frac{\delta_{\min}}{K+1} - \frac{3\Delta_{\min}}{4N^{(n)}(K+1)}\}$, $T_1 = \max\left\{\lceil \frac{32}{9} \frac{4^v(N^{(n)})^2 L_{\max}(\bar{r}-r)^2}{\Delta_{\min}^2 M_{\min}} \rceil, \lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \rceil\right\}$ or $T_1 = \max\left\{\lceil \frac{32}{9} \frac{4^v(N^{(n)})^2 (\bar{r}-r)^2}{\Delta_{\min}^2} \rceil, \lceil \frac{81}{32v} 16^v \rceil\right\}$ with edge information sharing, and $T_2 = \lceil N^{(n)}(L_{\max} + 1) + \frac{N^{(n)}L_{\max}\bar{r}}{\epsilon} \rceil$. The regret $\mathcal{R}(T)$ of D-DEBO is bounded $\mathcal{R}(T) \leq O(T^\zeta)$.

See Appendix D.2, available online for the proof. The underlying reason for restricting the last entering time is to impede a prohibitively large *exploitation regret* caused by the error-prone reward estimation of newly joined users.

Remark: The entering or leaving caused by user mobility can be phrased as shifting the original user-server assignment to a new stable one. Considering this, let us discuss another case which also leads to an update in the offloading scheme, i.e., computing capacity change. Every now and then, some edge servers may fail or resume to provide computing service due to shutdown, reboot, etc. Still suppose that server failure or restoration takes place at the beginning of each epoch with $K^{(n)}$ denoting the server number in epoch n . Note that L_{\max} is regarded as the potentially maximum resource a user can reach, which is assumed to be steady even the total computing capacity varies. Following the same vein, we present the regret result for the capacity change, where the proof is omitted since it is analogous to Theorem 3.

Proposition 2: Suppose that the epoch n' of the last server restoration satisfies $n' \leq O(\log_2 T^\zeta)$, $\zeta \in (0, 1)$. Let $\epsilon = \max\left\{\frac{\Delta_{\min}}{5N}, \frac{\delta_{\min}}{K^{(n)}+1} - \frac{3\Delta_{\min}}{4N(K^{(n)}+1)}\right\}$, $T_1 = \max\left\{\lceil \frac{32}{9} \frac{4^v N^2 L_{\max}(\bar{r}-r)^2}{\Delta_{\min}^2 M_{\min}} \rceil, \lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \rceil\right\}$ or $T_1 = \max\left\{\lceil \frac{32}{9} \frac{4^v N^2 (\bar{r}-r)^2}{\Delta_{\min}^2} \rceil, \lceil \frac{81}{32v} 16^v \rceil\right\}$ with edge information sharing, and $T_2 = \lceil N(L_{\max} + 1) + \frac{NL_{\max}\bar{r}}{\epsilon} \rceil$. The regret $\mathcal{R}(T)$ of D-DEBO is bounded $\mathcal{R}(T) \leq O(T^\zeta)$.

C. Fair Reward Distribution

So far, we have focused on the utilitarian compound reward optimization, which may cause *unfair server resource allocation*. Because users with higher rewards are more likely to be assigned to servers with faster transmission rate and processing speed, thereby blocking other users from the “better” edge servers. To address this issue, we explore the α -fairness maximization to enable a fair reward distribution among users, by changing the objective of OAP in (4) to $\max \sum_{i=1}^N u_\alpha(\mu_{ia_i})$ [24], where $u_\alpha(\mu_{ia_i})$ is defined:

$$u_\alpha(\mu_{ia_i}) = \begin{cases} \ln(1 + \beta\mu_{ia_i}) & \text{if } \alpha = 1, \\ \frac{\mu_{ia_i}^{1-\alpha}}{1-\alpha} & \text{if } \alpha \in (0, 1). \end{cases} \quad (13)$$

When $\alpha = 1$, it amounts to the proportional fairness. Slightly different from the standard proportional fairness, we adjust conventional $\ln \mu_{ia_i}$ to $\ln(1 + \beta\mu_{ia_i})$, $\beta > 0$ so as to enable the same fairness effect while further enhancing the model adaptability [25], [26]. Besides, we confine the value $\alpha \in (0, 1]$ rather than $\alpha \in (0, \infty]$ in [24] as $u_\alpha(\mu_{ij})$ would be negative if $\alpha > 1$, which is infeasible in the task offloading situation. Consistent with (13), let \mathbf{a}^f be the optimal assignment to the egalitarian fairness optimization, and define the fairness regret as $\mathcal{R}^f(T) = T \sum_{i=1}^N u_\alpha(\mu_{ia_i^f}) - \sum_{t=1}^T \sum_{i=1}^N u_\alpha(\mathbb{E}[r_{ia_i(t)}(t)])$.

We can still employ the DEBO in Algorithm 1 by replacing the reward estimations with their fairness values. Nevertheless, this operation is built on the premise that the fairness $u_\alpha(\tilde{r}_{ij}^{(n)})$ is sufficiently accurate if the estimated reward $\tilde{r}_{ij}^{(n)}$ converges to the expected reward μ_{ij} .

Lemma 5: Suppose the estimation error in RO Algorithm 2 satisfies $|\tilde{r}_{ij}^{(n)} - \mu_{ij}| \leq \Delta, \forall i \in \mathcal{N}, j \in \mathcal{K}_i$. For $\alpha = 1$, the fairness error fulfills

$$|u_\alpha(\tilde{r}_{ij}^{(n)}) - u_\alpha(\mu_{ij})| \leq \frac{\beta\Delta}{1 + \beta r}. \quad (14)$$

For $\alpha \neq 1$, the fairness error satisfies

$$|u_\alpha(\tilde{r}_{ij}^{(n)}) - u_\alpha(\mu_{ij})| \leq \frac{\Delta}{r^\alpha}. \quad (15)$$

See Appendix E, available online for the proof. Based on this lemma, we adapt the DEBO to a fair decentralized offloading F-DEBO, where the input to DAuction is $u_\alpha(\tilde{r}_{ij}^{(n)})$. Also, denote $\Delta_{\min}^f = \min_{\mathbf{a} \neq \mathbf{a}^f} \{\sum_{i=1}^N u_\alpha(\mu_{ia_i^f}) - \sum_{i=1}^N u_\alpha(\mu_{ia_i})\}$ and $\delta_{\min}^f = \min_{i \in \mathcal{N}} \min_{j, j' \in \mathcal{K}_i \cup \{K'\}, j \neq j'} \{|u_\alpha(\mu_{ij}) - u_\alpha(\mu_{ij'})|\}$. Theorem 4 presents the fairness regret, while the proof is omitted since it is similar to Theorem 1.

Theorem 4: Let $\epsilon = \max\{\frac{\Delta_{\min}^f}{5N}, \frac{\delta_{\min}^f}{K+1} - \frac{3\Delta_{\min}^f}{4N(K+1)}\}$, and $T_2 = \lceil N(L_{\max} + 1) + \frac{NL_{\max}u_\alpha(\bar{r})}{\epsilon} \rceil$. Set $T_1 = \max\{\lceil \frac{32}{9} \frac{4^v N^2 L_{\max}(\bar{r}-r)^2 \beta^2}{(\Delta_{\min}^f)^2 M_{\min}(1+\beta r)^2} \rceil, \lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \rceil\}$ or $T_1 = \max\{\lceil \frac{32}{9} \frac{4^v N^2 (\bar{r}-r)^2 \beta^2}{(\Delta_{\min}^f)^2 (1+\beta r)^2} \rceil, \lceil \frac{81}{32v} 16^v \rceil\}$ with edge information sharing when $\alpha = 1$, and $T_1 = \max\{\lceil \frac{32}{9} \frac{4^v N^2 L_{\max}(\bar{r}-r)^2}{(\Delta_{\min}^f)^2 M_{\min} r^\alpha} \rceil, \lceil \frac{81}{32v} \frac{16^v L_{\max}^2}{M_{\min}^2} \rceil\}$ or $T_1 = \max\{\lceil \frac{32}{9} \frac{4^v N^2 (\bar{r}-r)^2 \beta^2}{(\Delta_{\min}^f)^2 r^\alpha} \rceil, \lceil \frac{81}{32v} 16^v \rceil\}$ with edge information sharing when $\alpha \in (0, 1)$. The fairness regret

$\mathcal{R}^f(T)$ of F-DEBO is bounded:

$$\begin{aligned} \mathcal{R}^f(T) &\leq T_1 N u_\alpha(\bar{r}) \log_2(T+2) \\ &\quad + T_2 N u_\alpha(\bar{r}) \log_2(T+2) \\ &\quad + 12 N^2 K_{\max} u_\alpha(\bar{r}) \\ &= O(\log_2 T). \end{aligned} \quad (16)$$

Remark: The reward fairness among users in MAB is also studied in [18] through maximizing the product of all users' rewards in a centralized manner. Actually, our scheme can achieve the same fairness goal yet in a decentralized fashion, since taking a logarithm on the product is equivalent to maximizing the proportional fairness in (13).

V. HETEROGENEOUS TASK OFFLOADING SCHEME

This section explores the task offloading under the heterogeneous resource requirement, where server capacity stands for its endowed computing resource. Different from OAP, H-OAP is an APX-hard GAP problem with no optimal solution in polynomial time [15]. We will develop a new decentralized learning to handle users' heterogeneous requests.

A. Offline Problem Revisit

In contrast to OAP, one can only attain at most $(1 + \alpha)$ -approximate assignment \mathbf{a} to H-OAP, i.e., $\sum_{i=1}^N \mu_{ia_i} \geq \frac{1}{1+\alpha} \sum_{i=1}^N \mu_{ia_i^*}$, where $\alpha \geq 1$ is the approximation ratio to the following knapsack sub-problem:

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{N}_j} \mu_{ij} \\ \text{s.t.} \quad & C_j' \leq C_j. \end{aligned} \quad (17)$$

In fact, $\alpha = 1$ implies deriving the optimal solution to (17), and that is why we set the benchmark for the regret definition in (7) as $\frac{1}{2} \sum_{i=1}^N \mu_{ia_i^*}$. Besides, there is no capacity constraint for the virtual server K' .

To obtain a 2-approximate assignment, we first decouple the knapsack sub-problems, then acquire the optimal solution to each sub-problem sequentially. For this purpose, we introduce an indicator $\mathbf{I} = \{I_i, i \in \mathcal{N}\}$ to record the *current assignments* of all users, say $I_i = j$ means user i will offload tasks to server j , so as to enable the decoupling of $(K+1)$ sub-problems, including the virtual edge server also. Specifically, initialize each user as unassigned $\mathbf{I} = \mathbf{0}$, and define the reward $\Delta\mu_{ij}$ to mark the performance improvement if user i 's offloaded task is processed by another edge server:

$$\Delta\mu_{ij} = \begin{cases} \mu_{ij} & \text{if } I_i = 0, \\ \mu_{ij} - \mu_{ij'} & \text{if } I_i = j'. \end{cases} \quad (18)$$

According to [15], we solve (17) corresponding to each server j by using $\Delta\mu_{ij}, i \in \mathcal{N}$, i.e., gradually improve the offloading reward when allocating servers' computing resource. If the solution to (17) results in user i being assigned to server j , then update $I_i = j$ and recalculate $\Delta\mu_{ij}$ when dealing with the next

sub-problem for server $j \rightarrow j + 1$. As a result, the final indicator I is guaranteed to be a 2-approximate assignment to H-OAP if one can *optimally tackle all* $(K + 1)$ sub-problems. Considering the knapsack sub-problem is NP-hard, we implement a branch-and-bound method to efficiently search its optimal solution for ensuring $\alpha = 1$ [27].

B. Heterogeneous Decentralized Offloading

1) *Algorithm Design*: The analysis of solving H-OAP provides us a guide for designing dynamic offloading under the heterogeneous requirement. Similar to DEBO using learned rewards to attain the user-server assignment because of the unknown system-side information Θ , we also utilize the *reward estimation* $\tilde{r}^{(n)}$ along with local processing rewards $\mu_{K'}$ to deduce this assignment following the characterized indicator decoupling approach. Concretely, we show the heterogeneous decentralized epoch based offloading (H-DEBO) in Algorithm 4, with each epoch also consisting of an exploration phase, matching phase and exploitation phase. Denote $\bar{M}_{\min} = \frac{\min_{j \in \mathcal{K}} \{C_j\}}{\max_{i \in \mathcal{N}} \{c_i\}}$ as the ratio between the minimum server capacity and maximum user requirement.

2) *Exploration Phase*: Due to users' heterogeneous requests, random offloading by holding a specific resource unit in RO (Line 4 in Algorithm 2) is no longer applicable, since it is based on equally splitting the server capacity into multiple units. Therefore, we propose a *group offloading* scheme which requires each user to send a task to server in a *round-robin* fashion for reward exploration, where the group size is \bar{M}_{\min} so as not to violate any server capacity. As any user i can only reach servers $\mathcal{K}_i \cup K'$, the task will be locally handled (offloaded to virtual server) when the user is not covered by a server $j, j \in \mathcal{K}$ during the round-robin exploration. Lines 5-7 are for group number $N_g \leq K$ while Lines 9-13 amount to $N_g > K$. In particular, the estimated rewards only entail offloading to servers \mathcal{K} and exclude the virtual server even though tasks may be handled locally (choose K'). The exploration phase lasts for exactly T_1 time slots, i.e., T_1 observations with at least $\lfloor T_1/K \rfloor$ samples from each server for reward estimation $r^{(n)}$. Note that the group offloading remains decentralized as any user i can determine its group only based on local user index i . The computational complexity for each user is only $O(1)$ in every time slot.

Knowledgeable readers may ask why not each user directly offloads tasks to the reachable servers to learn the rewards, which can obviously reduce the exploration length. The main reasons lie in twofold. First, users are agnostic of each other, nor do they know the rest users covered by their reachable servers, and hence they have to explore longer to attain sufficient reward observations. Second, different users may reach different server subsets, which hinders the exploration coordination by offloading tasks in parallel, thus a round-robin reward exploration is essential.

3) *Matching and Exploitation Phases*: Similar to previous analysis, matching phase mainly entails solving the knapsack sub-problem of (17) successively. To this end, users will compute the rewards $\Delta \tilde{r}_{ij}^{(n)}, \forall j \in \mathcal{K}_i$, which are sent to each server for updating the indicator I (i.e., offloaded task is processed) based

Algorithm 4: H-DEBO: Heterogeneous Decentralized Epoch Based Offloading.

Inpit: $\{C_j, j \in \mathcal{K}\}, \{c_i, i \in \mathcal{N}\}, \bar{M}_{\min}, K, T_1$
1: Initialization: Set estimated rewards
 $\tilde{r}^{(n)} = \{\tilde{r}_{ij}^{(n)} = 0, \forall i \in \mathcal{N}, j \in \mathcal{K}_i\};$
2: **for** epoch $n = 1$ to n_T **do**
3: Users form N_g groups with group size being \bar{M}_{\min} ;
 \triangleright **Start Exploration Phase**
4: **for** $t = 1$ to T_1 **do**
5: **if** $N_g \leq K$ **then**
6: Users in group k offload tasks to server
 $[(t-1)\%K + k]\%(K+1)$, if unreachable,
choose K' ;
7: Update $\tilde{r}_{ij}^{(n)}, j \in \mathcal{K}_i$ using $r_{ij}(t), j \in \mathcal{K}_i$;
8: **else**
9: **for** $g = 1$ to $\lfloor N_g/K \rfloor$ **do**
10: Users in group $k, k = (g-1)K + 1, \dots, gK$
offload tasks to server
 $[(t-1)\%K + k - (g-1)K]\%(K+1)$, if
unreachable, choose K' ;
11: Update $\tilde{r}_{ij}^{(n)}, j \in \mathcal{K}_i$ using $r_{ij}(t), j \in \mathcal{K}_i$;
12: Users in group $k, k = \lfloor N_g/K \rfloor K + 1, \dots, N_g$
offload tasks to server
 $[(t-1)\%K + k - \lfloor N_g/K \rfloor K]\%(K+1)$, if
unreachable, choose K' ;
13: Update $\tilde{r}_{ij}^{(n)}, j \in \mathcal{K}_i$ using $r_{ij}(t), j \in \mathcal{K}_i$;
14: Initialize the indicator vector $I = \{I_i = 0, i \in \mathcal{N}\}$;
 \triangleright **Start Matching Phase**
15: **for** edge server $j \in \mathcal{K}$ **do**
16: **if** edge server $j \in \mathcal{K}_i$ **then**
17: Each user i computes $\Delta \tilde{r}_{ij}^{(n)}$ similar to (18),
offloads a task and sends $\Delta \tilde{r}_{ij}^{(n)}$ to server j ;
18: Server j performs branch-and-bound to solve (17)
with the input $\{\Delta \tilde{r}_{ij}^{(n)}, i \in \mathcal{N}\}$;
19: Each user i updates $I_i = j$ if assigned to server j ;
20: **else**
21: Each user i offloads a task to virtual server K' ;
22: Each user i updates $I_i = K'$ if $\mu_{iK'} > \tilde{r}_{iI_i}^{(n)}$ or $I_i = 0$;
23: Set $\mathbf{a}' = I$; \triangleright Assignment from matching
24: **for** remaining 2^n time slots **do**
25: Each user i offloads tasks to the assigned server a'_i ;
 \triangleright **Exploitation Phase**

on branch-and-bound method, that is offloading tasks to servers according to the results in Lines 18-19. Otherwise they will process the tasks themselves, but the indicator I will not be renewed (Lines 20-21). After offloading tasks to all servers \mathcal{K} , users finally determine the assignments by comparing with their local execution (Line 22). If every estimated reward $\tilde{r}_{ij}^{(n)}$ closely approaches the expected reward μ_{ij} , the obtained \mathbf{a}' will be a 2-approximate assignment. Also, the matching phase lasts for $(K + 1)$ time slots corresponding to $(K + 1)$ (virtual) servers. Each user has $O(1)$ computation to obtain $\Delta \tilde{r}_{ij}^{(n)}$ and each server incurs $O(NC_j)$ to run branch-and-bound, then total computation is $O((T_1 + NC_j K) \log_2 T)$ due to at most $O(\log_2 T)$ epochs.

Finally, users will exploit the assignment \mathbf{a}' for 2^n time slots, while other dominated exploitation length enables the same effect.

C. Performance Analysis of H-DEBO

Now we analyze the regret $\tilde{\mathcal{R}}(T)$ in (7). Mark reachable server intersection as $\mathcal{K}_{i,i'} = \mathcal{K}_i \cap \mathcal{K}_{i'}$. Let $\delta_{\min}^{(1)} = \min_{i,i' \in \mathcal{N}, i \neq i'} \min_{j \in \mathcal{K}_{i,i'} \cup \{K'\}, k \in \mathcal{K}_{i'} \cup \{K'\}, j \neq k} |\mu_{ij} - (\mu_{i'j} - \mu_{i'k})|$, $\delta_{\min}^{(2)} = \min_{i,i' \in \mathcal{N}, i \neq i'} \min_{j \in \mathcal{K}_{i,i'} \cup \{K'\}, j' \in \mathcal{K}_i \cup \{K'\}, k \in \mathcal{K}_{i'} \cup \{K'\}, j \neq j', j' \neq k} |(\mu_{ij} - \mu_{ij'}) - (\mu_{i'j} - \mu_{i'k})|$, $\delta_{\min} = \min_{i \in \mathcal{N}} \min_{j, j' \in \mathcal{K}_i \cup \{K'\}, j \neq j'} \{|\mu_{ij} - \mu_{ij'}|\}$. Denote $\delta'_{\min} = \min\{\delta_{\min}^{(1)}, \delta_{\min}^{(2)}, \delta_{\min}\}$, $\bar{M}_{\max} = \frac{\max_{j \in \mathcal{K}} \{C_j\}}{\min_{i \in \mathcal{N}} \{c_i\}}$ and $K_{\min} = \min_{i \in \mathcal{N}} \{K_i\}$. The following theorem states the regret with its proof being presented in Appendix F, available online.

Theorem 5: Let $T_1 = \left\lceil \frac{25\bar{M}_{\max}^2(\bar{r}-r)^2 K}{2(\delta'_{\min})^2} \right\rceil$ or $T_1 = \left\lceil \frac{25\bar{M}_{\max}^2(\bar{r}-r)^2 K}{2(\delta'_{\min})^2 K_{\min}} \right\rceil$ with edge information sharing if $N_g \leq K$, and $T_1 = \left\lceil \frac{25\bar{M}_{\max}^2(\bar{r}-r)^2}{2(\delta'_{\min})^2} (K+N) \right\rceil$ or $T_1 = \left\lceil \frac{25\bar{M}_{\max}^2(\bar{r}-r)^2 (K+N)}{2(\delta'_{\min})^2 K_{\min}} \right\rceil$ with edge information sharing if $N_g > K$. The regret $\tilde{\mathcal{R}}(T)$ of H-DEBO is upper bounded by

$$\tilde{\mathcal{R}}(T) \leq \left[T_1 \frac{N\bar{r}}{2} + (K+1) \frac{N\bar{r}}{2} \right] \log_2(T+2) + 4N^2 K_{\max} \bar{r} = O(\log_2 T). \quad (19)$$

Remark: All extensions in Section IV can be applied to H-DEBO, which are omitted due to page limit. Akin to DEBO, we could adjust δ'_{\min} to a larger value in practice to prevent too large T_1 . Besides, the accumulated rewards may exceed $\frac{T}{2} \sum_{i=1}^N \mu_{ia_i^*}$ which is in fact a theoretically lower bound.

VI. PERFORMANCE EVALUATION

In this section, we show the evaluated results of proposed decentralized offloading schemes and baseline methods.

A. Evaluation Setup

1) *Parameter Setting:* Consider the MEC system (divided into cells) is composed of $K = 3$ to $K = 5$ edge servers and $N = 8$ to $N = 12$ mobile users. In line with the real measurements [13], task size b_i is distributed in [500,1600] KB with required CPU cycles per bit being $\gamma_i = 1000$. Server processing speed f_j is in [4,8] GHz, and cellular transmission rate s_j is set in [9,11] Mbps according to the typical 4 G uplink speed [28]. The CPU speeds of mobile devices are in [1,2] GHz. Server capacity for maximum task service M_j is an integer in [1,4], while for endowed computing resource C_j is randomly in [2,2.5] with user resource requirement $c_i \in [0.5, 1]$. The reward preference function of (3) is $\mu_{ij} = v_i - \rho_i d_{ij}$ with task value $v_i \in [3, 3.5]$ and $\rho_i \in [0.2, 0.5]$. The random reward observation $r_{ij}(t) \in [\mu_{ij} - 0.3, \mu_{ij} + 0.3]$, and also let $r = 0.3$, $\bar{r} = 3.8$ accordingly. The total number of time slots is $T = 6 \times 10^6$.

2) *Benchmark Algorithms:* To show the effectiveness of our proposed DEBO, its extensions and H-DEBO, we introduce the following algorithms as benchmarks for comparison.

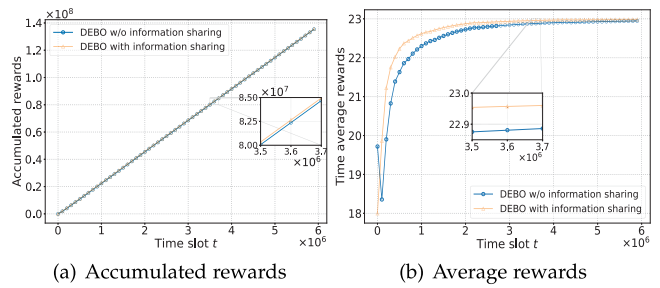


Fig. 2. Influence of information sharing on rewards.

- DM-Non0: decentralized multi-user MAB with non-zero rewards [31]. To the best of our knowledge, this is the only work that studies non-collision model, where the reward of any individual user also depends on the user number playing the same arm. As original DM-Non0 mainly applies to the case where any user can get access to all arms (all edge servers), we adapt it to our problem that each user can only pull partial arms (reachable servers).
- M-UCB: upper confidence bound (UCB) which pulls arm with the highest confidence bound of the average reward [29]. Considering there are multiple users, suppose that they independently execute UCB for task offloading, namely M-UCB.
- M-EXP3: exponential-weight algorithm for exploration and exploitation (EXP3) that pulls arms following the exponential-weight probability [30]. For our problem, multiple users locally choose edge servers based on EXP3, i.e., M-EXP3.

The optimal assignment \mathbf{a}^* to OAP is obtained through Hungarian algorithm [32] by splitting each server j into M_j copies, and is utilized to compute the regret $\mathcal{R}(T)$ in (5).

B. Influence of Edge Information Sharing

Information sharing among edge servers enables a better reward exploration. For brevity, we mainly evaluate DEBO without and with such information exchange, respectively, given edge server $K = 4$ and mobile user $N = 10$.

Accumulated/Average Rewards: We show the accumulated and time average rewards of DEBO in Fig. 2. The results indicate that the edge information sharing facilitates a more accurate estimated rewards within a shorter exploration, thereby improving the accumulated and time average rewards. Note that DEBO still achieves favorable performance with a time average reward of 22.948 even without neighboring edge feedback, while that for the sharing case is 22.983, slightly higher indeed.

Accumulated/Average Regret and Ratio: Also, we exhibit the accumulated/average regrets when information sharing exists or not in Fig. 3(a) and (b). Consistent with the reward observation, DEBO with edge information exchange yields lower regrets than the non-sharing case. We then display the ratio between time average rewards and the optimal result in Fig. 3(c), where both cases attain a near to optimal result.

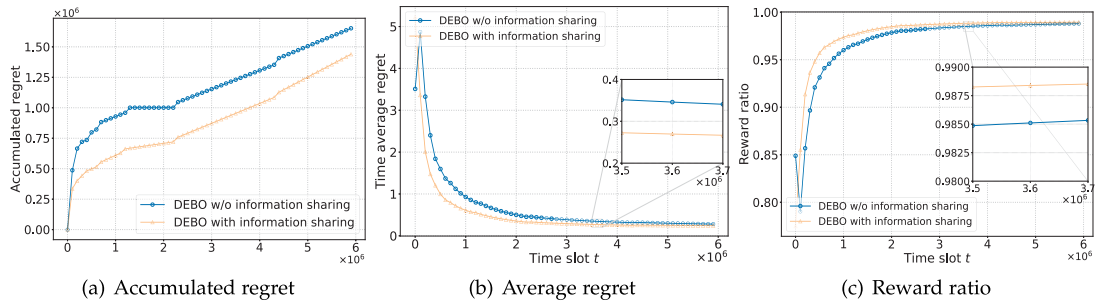


Fig. 3. Offloading regret and reward ratio over time with and w/o sharing.

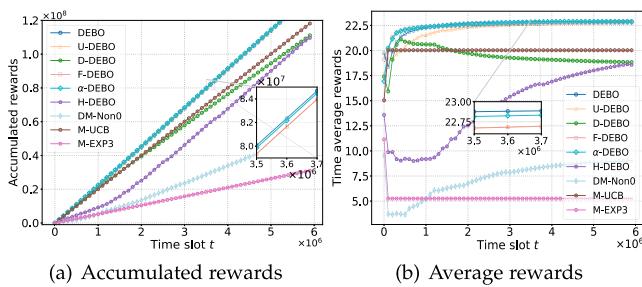


Fig. 4. Offloading rewards over time.

To avoid redundancy, we next evaluate the performance of DEBO, its extensions and baseline methods *mainly considering the case without edge information sharing*.

C. Evaluation Results Over Time

We carry out the evaluations given $K = 4$ and $N = 10$. In particular, we simulate both proportional fairness ($\alpha = 1$) and the fairness case with $\alpha = 0.5$, where the legend of the former is F-DEBO and that of the latter is α -DEBO, respectively. Later regret results of F-DEBO and α -DEBO are computed by using the fairness regret metric $\mathcal{R}^f(T)$.

Accumulated/Average Rewards: The accumulated and time average rewards of our proposed and baseline algorithms are shown in Fig. 4. One can see that our algorithms outperform the benchmarks as we can always achieve higher accumulated and time average rewards. Particularly, DEBO yields the highest rewards since more information is assumed known in advance, meanwhile U-DEBO, F-DEBO and α -DEBO also have satisfactory performances with comparable rewards to DEBO. As for D-DEBO, we consider that a user will enter or leave the MEC system with a certain probability at the beginning of each epoch. D-DEBO leads to slightly lower rewards mainly because dynamic user leaving will cause a reduction in the compound reward $\sum_{i=1}^{N(t)} \mu_i \alpha_i$. The average rewards of DEBO and its extensions will stabilize over time, and approach the optimal results, respectively. Moreover, H-DEBO actually attains favorable outcome with large accumulated and average rewards. DM-Non0 has lower rewards since it is originally designed for the case where all servers can be reached. The baseline method M-UCB has inferior performance to our proposed decentralized schemes

due to the lack of coordination among users. M-EXP3 yields the lowest rewards owing to stochastic task offloading. Therefore, strawman extension of single-user UCB and EXP3 to multi-user scenarios is insufficient to obtain satisfactory performances.

Accumulated/Average Regret and Ratio: The accumulated and time average regrets are shown in Fig. 5(a) and (b), where H-DEBO is excluded as we can not derive the optimal assignment to H-OAP. Note that our proposed algorithms give rise to orderly lower regrets than the benchmarks. The regrets of DEBO, U-DEBO, F-DEBO and α -DEBO are negligibly small. Also, the regret of D-DEBO is very low, which is because the optimal assignment will be changing when we have dynamic user entering or leaving. The reward ratios of all algorithms are displayed in Fig. 5(c), which indicate that the ratios of DEBO and its extensions will converge to steady values close to 1, namely their regrets are sub-linear in terms of the total time horizon T . As for DM-Non0, M-UCB and M-EXP3, there exist performance gaps between their ratios and 1. Though the setting of D-DEBO is different from other algorithms due to the dynamic environment, here we show its results together with others' not for comparison but mainly to demonstrate its sub-linear regrets and save space.

D. Evaluation Results Over User Number

Varying the number of users N from 8 to 12 while keeping the server number $K = 4$ unchanged, we obtain the time average rewards and regret in each case.

Rewards, Regret and Ratio: The time average rewards, regret and reward ratio over user number N are shown in Figs. 6, 7, and 8, respectively. With the increase of N , the rewards of all algorithms will tend to increase. Besides, our proposed DEBO, U-DEBO, F-DEBO, α -DEBO and H-DEBO always achieve better performances than baseline methods DM-Non0, M-UCB and M-EXP3 pertaining to the rewards, regret and reward ratio.

Fairness Demonstration: F-DEBO and α -DEBO are proposed to ensure a more equitable reward distribution among users by introducing the α -fairness. Previously, we have already shown that they have comparable rewards to DEBO. Fig. 9 further displays their maximum gaps of users' time average rewards. We can observe that the gap (fairness) is significantly reduced (enhanced) owing to the fairness consideration. In

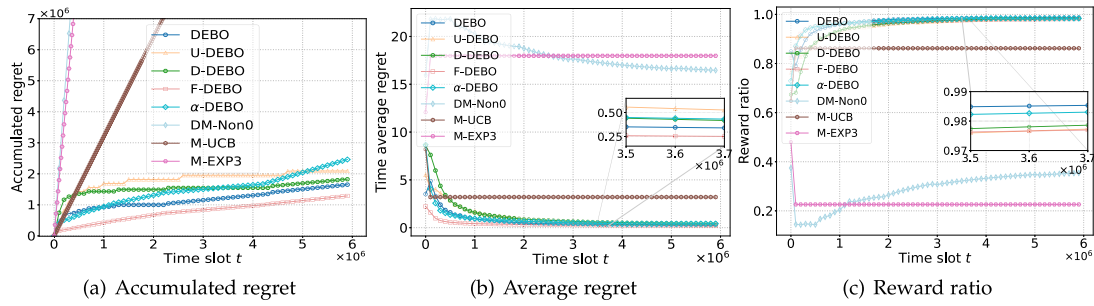


Fig. 5. Offloading regret and reward ratio over time.

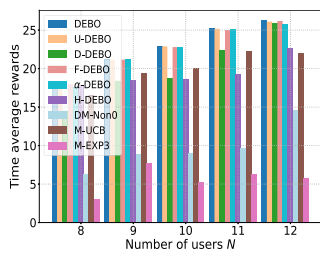


Fig. 6. Rewards versus N .

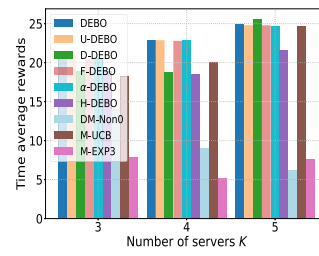


Fig. 10. Rewards versus K .

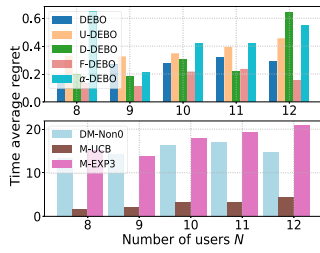


Fig. 7. Regret versus N .

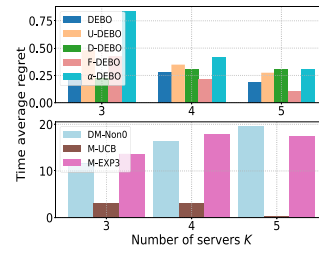


Fig. 11. Regret versus K .

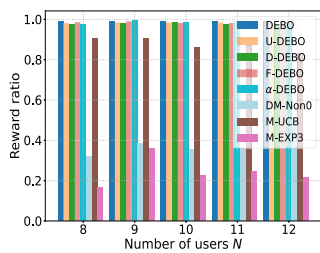


Fig. 8. Reward ratio versus N .

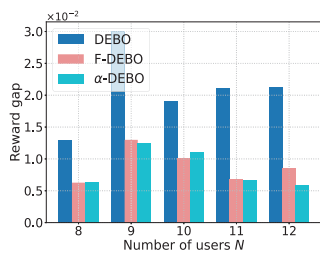


Fig. 9. Reward gap versus N .

particular, F-DEBO (α -DEBO) accomplishes 56.54% (58.52%) fairness improvement with only sacrificing 0.49% (0.97%) compound reward on average.

E. Evaluation Results Over Server Number

We last explore the influence of the server number by varying K from 3 to 5 given fixed user number $N = 10$.

Rewards, Regret and Ratio: Similarly, we exhibit the time average rewards, regret and reward ratio in Figs. 10–12, respectively. When K becomes large, rewards of all methods will increase accordingly, while our proposed DEBO, U-DEBO, F-DEBO, α -DEBO and H-DEBO consistently attain better performance than benchmarks in terms of the rewards, regret and ratio. Surprisingly, M-UCB also achieves near-optimal result when $K = 5$. This is because users rarely encounter any collisions when total server capacity is adequate, i.e., the multi-user offloading indeed degenerates to single-user task placement. In

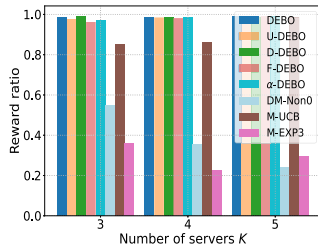
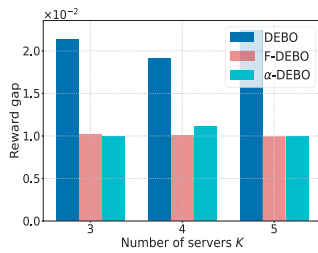
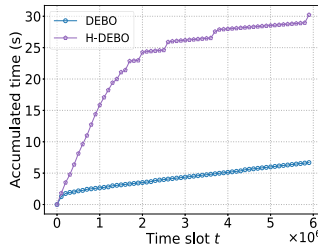
Fig. 12. Reward ratio versus K .Fig. 13. Reward gap versus K .

Fig. 14. Running time.

contrast, DM-Non0 and M-EXP3 still perform poor with low rewards and high regrets.

Fairness Demonstration: According to Fig. 13, F-DEBO and α -DEBO still enhance the fairness of server resource allocation to mobile users. More concretely, F-DEBO (α -DEBO) achieves 51.75% (50.39%) fairness increase with merely 1.51% (1.37%) rewards reduction.

F. Extended Results

Lastly, we extend previous results to further involve computational complexity, task delay, and larger network scale.

Computation and Delay: As aforementioned, we have theoretically illustrated that the computational complexity of DEBO and H-DEBO is low indeed. To verify this, we display their running time when $N = 10, K = 4$ using an Intel i7-11 Windows machine. From Fig. 14, we observe that the accumulated running time is sub-linear over T , which is several or tens of seconds, respectively. Moreover, we show the time average task delay of our proposed and baseline methods in Fig. 15. Even we aim to optimize the accumulated rewards, we can see that our algorithms still have lower task delay than benchmarks. Note that DM-Non0 has smaller delay at the beginning mainly because

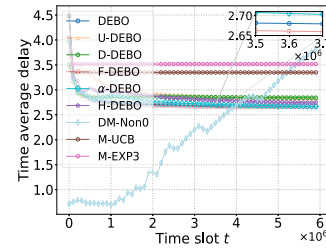


Fig. 15. Task delay.

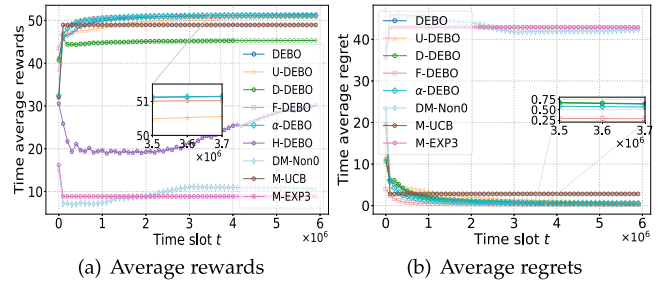


Fig. 16. Results of large-scale MEC.

there is a long period of group offloading without violating any server capacity.

Large MEC Network: We finally validate the scalability of our offloading schemes to a larger MEC network. Specifically, we show the time average rewards and regret in Fig. 16 when the user number $N = 20$ and server number $K = 8$. We can see that DEBO, U-DEBO, F-DEBO, α -DEBO and H-DEBO also perform better than benchmarks, especially pertaining to the regret. Though M-UCB has relatively high rewards, it is still sub-optimal as its regret is large. Overall, our algorithms can be extended to a larger MEC network.

VII. RELATED WORK

In this article, we study the decentralized task offloading in a dynamic MEC system with unknown system-side information. Let us briefly survey the related works.

MEC Task Offloading: Emerging MEC enables users to offload their computation-intensive tasks to local edge servers [1]. Chen et al. propose a response updating method to make the offloading decision given the full system-side information [9]. Considering system uncertainty, Ouyang et al. explore the user-managed service placement by formulating task offloading as a contextual MAB problem, which mainly focuses on the single-user case [8]. Following this line, Zhang et al. study centralized task offloading under undisclosed task reward and limited server capacity, while they allow a certain degree of capacity violation [33]. Mitsis et al. explore the task offloading for users and service pricing for edge servers, where each edge server employs reinforcement learning, especially the UCB policy, to attain the optimal service price [34]. Wu et al. propose a multi-user MAB method to solve the task offloading problem in a two-tier edge computing system, including mobile users, edge servers and core cloud. However, they do not consider limited server

capacity since edge servers can further offload computation to the cloud [35]. To improve the offloading efficiency in the edge computing, a delay-optimal cooperative mobile edge caching scheme is proposed to enhance the MEC service quality with low complexity [36]. Similarly, authors in [37] consider the service caching and propose a novel service-oriented network slicing approach to efficiently manage the multi-dimensional network resource. So far, decentralized multi-user task offloading with limited computing capacity and bandit feedbacks still remains open.

Decentralized Multi-User MAB: MAB is a representative model for the sequential decision making, where a decentralized framework is developed in [17] for communication based multi-user MAB. Bistritz et al. first characterize a fully decentralized bandit policy with heterogeneous rewards based on the theory of unperturbed chain [10]. Later works further apply this framework to the wireless channel allocation [11]. However, these existing researches are built on the collision-based reward model, which actually allows an indirect communication signal for decentralized coordination. To the best of our knowledge, only [31] exploits non-zero rewards even collision occurs, but the reward depends on the user number making the same action as long as the number is under a uniform predefined value for all arms.

Fairness in MAB: Fairness issue in online learning has attracted a surge of interests. Nevertheless, previous works mainly study arm fairness in single-user scenarios where the selection probability of each arm is to be higher than a predefined proportion [38], or multi-user reward fairness through centralized decision making [18]. A collision based multi-user reward fairness is also explored in [22].

VIII. CONCLUSION

In this article, we study a fully decentralized task offloading for a dynamic MEC system with the unknown system-side information and overlapping server coverage. We divide the time horizon into epochs and propose DEBO considering both cases without and with edge information sharing. For both cases, DEBO can ensure an $O(\log_2 T)$ regret of the dynamic task offloading in a decentralized manner. On this basis, we also extend DEBO to handle situations such as the unknown reward gap, dynamic user entering or leaving, and fair reward distribution, where sub-linear regrets are obtained for all extensions. Incorporating users' heterogeneous resource requests, we further develop H-DEBO which achieves an $O(\log_2 T)$ regret and satisfactory offloading rewards. Extensive evaluations show the superiority of our proposed algorithms compared to existing benchmarks.

REFERENCES

- [1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
- [2] R. Kelly, "Internet of Things data to top 1.6 Zettabytes by 2020," *Campus Technol.*, vol. 9, pp. 1536–1233, 2016.
- [3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
- [5] Y. Li, H. C. Ng, L. Zhang, and B. Li, "Online cooperative resource allocation at the edge: A privacy-preserving approach," in *Proc. IEEE 28th Int. Conf. Netw. Protoc.*, 2021, pp. 1–11.
- [6] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 37–45.
- [7] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 1, pp. 2637–2646, Nov. 2017.
- [8] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1468–1476.
- [9] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 748–756.
- [10] I. Bistritz and A. Leshem, "Distributed multi-player bandits—a game of thrones approach," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7222–7232.
- [11] S. M. Zafaruddin, I. Bistritz, A. Leshem, and D. Niyato, "Distributed learning for channel allocation over a shared spectrum," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2337–2349, 2019.
- [12] S. J. Darak and M. K. Hanawal, "Multi-player multi-armed bandits for stable allocation in heterogeneous ad-hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2350–2363, Oct. 2019.
- [13] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.
- [14] Y. Jararweh, A. Doulat, O. Al-Qudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun.*, 2016, pp. 1–5.
- [15] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Inf. Process. Lett.*, vol. 100, no. 4, pp. 162–166, 2006.
- [16] H. Tibrewal, S. Patchala, M. K. Hanawal, and S. J. Darak, "Distributed learning and optimal assignment in multiplayer heterogeneous networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1693–1701.
- [17] N. Nayyar, D. Kalathil, and R. Jain, "On regret-optimal learning in decentralized multiplayer multiarmed bandits," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 597–606, Mar. 2018.
- [18] S. Hossain, E. Micha, and N. Shah, "Fair algorithms for multi-agent multi-armed bandits?" *Adv. Neural Inf. Proc. Syst.*, vol. 34, pp. 24005–24017, 2021.
- [19] D. P. Bertsekas and D. A. Castanon, "The auction algorithm for the transportation problem," *Ann. Operations Res.*, vol. 20, no. 1, pp. 67–96, 1989.
- [20] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.
- [21] T. Ouyang, X. Chen, Z. Zhou, R. Li, and X. Tang, "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1313–1326, Mar. 2023.
- [22] I. Bistritz, T. Z. Baharav, A. Leshem, and N. Bambos, "My fair bandit: Distributed learning of max-min fairness with multi-player bandits," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 930–940.
- [23] O. Naparstek and A. Leshem, "Fully distributed optimal channel assignment for open spectrum access," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 283–294, Jan. 2014.
- [24] H. Shi, R. V. Prasad, E. Onur, and I. G. M. M. Niemegeers, "Fairness in wireless networks: issues, measures and challenges," *IEEE Commun. Surveys Tut.*, vol. 16, no. 1, pp. 5–24, First Quarter 2014.
- [25] F. Kelly, "Charging and rate control for elastic traffic," *Eur. Trans. Telecommun.*, vol. 8, no. 1, pp. 33–37, 1997.
- [26] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 2420–2428.
- [27] R. E. Neapolitan and K. Naimipour, *Foundations of Algorithms Using C Pseudocode*. Boston, MA, USA: Jones and Bartlett, 2004.

- [28] 4G4U, “4 G speed tests,” Aug. 22, 2019, Accessed: May 19, 2023. [Online]. Available: <https://www.4g4u.org/4g-speed-tests/>
- [29] P. Auer, N. C. Bianchi, and P. Fischer, “Finite-time analysis of the multi-armed bandit problem,” *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
- [30] P. Auer, N. C. Bianchi, Y. Freund, and R. E. Schapire, “The nonstochastic multiarmed bandit problem,” *SIAM J. Comput.*, vol. 32, no. 1, pp. 48–77, 2002.
- [31] A. Magesh and V. V. Veeravalli, “Decentralized heterogeneous multi-player multi-armed bandits with non-zero rewards on collisions,” *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2622–2634, 2022.
- [32] H. W. Kuhn, “The hungarian method for the assignment problem,” *Nav. Res. Logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.
- [33] X. Zhang, R. Zhou, Z. Zhou, J. C. S. Lui, and Z. Li, “An online learning-based task offloading framework for 5G small cell networks,” in *Proc. 49th Int. Conf. Parallel Process.*, 2020, pp. 1–11.
- [34] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, “Price and risk awareness for data offloading decision-making in edge computing systems,” *IEEE Syst. J.*, vol. 16, no. 4, pp. 6546–6557, Dec. 2022.
- [35] B. Wu, T. Chen, and X. Wang, “Multi-agent multi-armed bandit learning for online management of edge-assisted computing,” *IEEE Trans. Commun.*, vol. 69, no. 12, pp. 8188–8199, Dec. 2021.
- [36] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, “Cooperative edge caching in user-centric clustered mobile networks,” *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.
- [37] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, “Air-ground integrated vehicular network slicing with content pushing and caching,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2114–2127, Sep. 2018.
- [38] F. Li, J. Liu, and B. Ji, “Combinatorial sleeping bandits with fairness constraints,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1702–1710.
- [39] T. L. Lai and H. Robbins, “Asymptotically efficient adaptive allocation rules,” *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.
- [40] 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 Conf. Comput. Commun.*, 2022, pp. 1199–1208.



Xiong Wang (Member, IEEE) received the BE degree in electronic information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2014, and the PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. He was a post-doctoral fellow with Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China, from 2019 to 2021. He is currently an associate professor in School of Computer Science and Technology, Huazhong University of Science and

Technology, Wuhan, China. His research interests include distributed machine learning system, federated learning, network flow control, mean field analysis, cloud/edge computing.



Jiancheng Ye (Member, IEEE) received the BE degree in network engineering from Sun Yat-sen University, Guangzhou, China, in 2008, the MPhil degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2011, and the PhD degree in computer networking from the University of Hong Kong, Hong Kong, in 2018. He was a software engineer with Harmonic Inc., from 2011 to 2014, and a post-doctoral fellow with The University of Hong Kong, from 2018 to 2019. His research interests include congestion

control, queue management, optimization of computer networks, edge computing, and online learning.



John C.S. Lui (Fellow, IEEE) received the PhD degree in computer science from the University of California, Los Angeles. He is currently the Choh-Ming Li chair professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include machine learning, online learning (e.g., multi-armed bandit, reinforcement learning), network science, future internet architectures and protocols, network economics, network/system security, large scale storage systems. He is an elected member of the IFIP WG 7.3,

Fellow of ACM, Senior Research Fellow of the Croucher Foundation and was the past chair of the ACM SIGMETRICS (2011-2015). He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. John is a co-recipient of the best paper award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP NOMS 2006, SIMPLEX 2013, and ACM RecSys 2017.