Decentralized Task Offloading in Edge Computing: A Multi-User Multi-Armed Bandit Approach

Xiong Wang*, Jiancheng Ye†, and John C.S. Lui‡
* 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, China
† Network Technology Lab and Hong Kong Research Center, Huawei Technologies Co., Ltd, Hong Kong
‡ Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
E-mail: xiongwang@hust.edu.cn, yejiancheng@huawei.com, cslui@cse.cuhk.edu.hk

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796961

***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 a dynamic and 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. 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 subject to the network delay. We show that DEBO can deduce the optimal user-server assignment, thereby achieving a *close-to-optimal* service performance and *tight* $O(\log T)$ offloading 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.**

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 5G 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 computing 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 and Wi-Fi access points. Under

This work is supported in part by the GRF 14200321 and CUHK:6905407. (Corresponding author: Jiancheng Ye.)

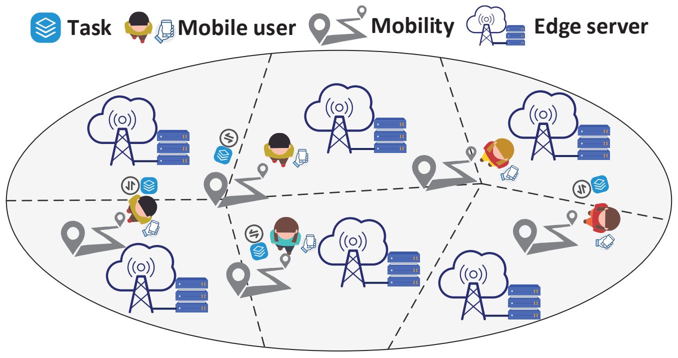


Fig. 1: MEC system with separate service regions.

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 radio coverage of edge servers as shown in Fig. 1, and this may lead to the performance disparity caused by user roaming across different service regions. Therefore, one of the core problems is to make *effective offloading decision* in order to meet users' stringent delay requirements and augment servers' 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, cellular bandwidth) 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]–[6], which will be ineffective when this information is unknown or dynamically changing. Few works address system uncertainty via online learning based offloading schemes for service augmentation [7]–[9]. However, they mainly focus on *centralized offloading* with task placement decided by a *single* 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 the 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 feedbacks. Though decentralized policies have been proposed [10]–[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 often have *various latency-sensitivity*, which inevitably leads to distinct offloading rewards. This requires balancing their delay-sensitive rewards to ensure a fair edge resource allocation. 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 an edge server is regarded as playing an arm. 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 deriving the optimal user-server assignment in a decentralized manner, only using historically perceived observations. Furthermore, we advocate a heterogeneous DEBO (H-DEBO) to accommodate users' heterogeneous offloading 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. Our scheme achieves optimal performance for coexisting users without any inter-user communication or system-side information. To the best of our knowledge, this is the *first* work that conducts a thorough analysis of decentralized offloading under system uncertainty.
- We propose DEBO to divide time horizon into epochs for dynamic task placement, where each epoch consists of an exploration, matching and exploitation phase to learn the optimal user-server assignment. DEBO attains a *tight* $O(\log T)$ offloading regret merely using *bandit* reward feedbacks.
- 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 the H-DEBO to accommodate *heterogeneous* offloading requirements, where an $O(\log T)$ regret is accomplished by solving an APX-hard assignment problem only with *learned* results.
- Extensive evaluations based on real measurements are performed to show the superiority of our offloading schemes over the existing approaches. In particular, we can achieve *51.49% fairness improvement* by only *sacrificing 0.44% rewards* when incorporating the fair reward distribution.

II. System Model

We consider an MEC system with a set of mobile users $\mathcal{N} = \{1, 2, ..., N\}$ and edge servers $\mathcal{K} = \{1, 2, ..., K\}$ which can provide computing services. When the system scales, users only retain local *user indices* and need to make offloading decisions *independently* as they are agnostic of each other.

A. User Task Offloading

Each user will offload a computation-intensive task to a server at any time $t \in \{1, 2, ..., T\}$ where the total time horizon T is *unspecified*. Suppose s_j and f_j are the 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]. 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, it will experience a delay d_{ij}, which includes transmission and processing time:

$$d_{ij} = \frac{b_i}{s_j} + \frac{b_i \gamma_i}{f_j}. \tag{1}$$

Along with the delay, user i also associates an instant reward μ_{ij} to represent its personal preference on the service performance:

$$\mu_{ij} = v_i - g_i(d_{ij}), \tag{2}$$

where v_i is the intrinsic task value and $g_i(\cdot)$ is a cost function which increases with 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 $\mu_{ij} = \mathbb{E}[r_{ij}(t)], r_{ij}(t) \in [\underline{r}, \overline{r}], \forall i \in \mathcal{N}, j \in \mathcal{K}$ where positive $\underline{r}$ and $\overline{r}$ are the lower and upper reward bounds.

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 resource (memory, CPU, storage) to process their offloaded tasks, the task service capacity M_j of server j represents how many tasks it can handle concurrently. If excessive computation arrives, 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 $r_{ij}(t) = 0$. W.l.o.g., assume $M = \sum_{j=1}^{K} M_j \geq N$, i.e., MEC system has adequate resource to fulfill all offloading requests.

2) Endowed Computing Resource: Considering users require heterogeneous resource for task offloading, say running different types of applications, the capacity C_j of edge server j stands for the amount of its endowed computing resource.

C. Problem Formulation

1) Homogeneous Offloading: Under the homogeneous requirement, computing capacity of each server implies the maximum number of admitted tasks. At time t, denote $a_i(t) \in \mathcal{K}$ as the selected server decision of user i with $\boldsymbol{a}(t) = \{a_i(t), i \in \mathcal{N}\}$, 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 $\boldsymbol{a}^*$ 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:

$$\begin{aligned} \max \ & \sum_{i=1}^{N} \mu_{ia_i} \\ \text{s.t. } & |\mathcal{N}_j| \leq M_j, \forall j \in \mathcal{K}. \end{aligned} \tag{3}$$

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

$$\mathcal{R}(T) = T \sum_{i=1}^{N} \mu_{ia_i^*} - \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{E}[r_{ia_i(t)}(t)]. \tag{4}$$

Note that $\mathbb{E}[r_{ia_i(t)}(t)] = \mu_{ia_i(t)}$ if the offloaded task is processed, otherwise $r_{ia_i(t)}(t) = 0$ once being discarded. For ease of exposition, we also refer to $\boldsymbol{a}^*$ or $\boldsymbol{a}(t)$ as an *assignment* between mobile users and edge servers.

2) Heterogeneous Offloading: 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 at time t. Again, the optimal offloading is $\boldsymbol{a}^*$, which can be obtained by solving the heterogeneous offline assignment problem (H-OAP) given the system-side information Θ:

$$\begin{aligned} \max \ & \sum_{i=1}^{N} \mu_{ia_i} \\ \text{s.t. } & C'_j(t) \leq C_j, \forall j \in \mathcal{K}. \end{aligned} \tag{5}$$

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 $\boldsymbol{a}$ 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., $\alpha = 1$:

$$\tilde{\mathcal{R}}(T) = \frac{T}{2} \sum_{i=1}^{N} \mu_{ia_i^*} - \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{E}[r_{ia_i(t)}(t)]. \tag{6}$$

If the task is rejected due to any server capacity limitation, we have $r_{ia_i(t)}(t) = 0$. Also, the offloading decisions or assignment $\boldsymbol{a}(t)$ 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 schemes due to the undisclosed system-side information and agnostic user coexistence. Another challenge is that because edge servers can accommodate multiple tasks, previous collision-based indirect collaboration frameworks are not applicable to our offloading design [10], [16]. Last but not least, APX-hard H-OAP has no optimal solution in polynomial time caused by heterogeneous 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.

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. Since the total time T is unspecified aperior, we divide the time horizon into epochs $\{1, 2, ..., n_T\}$, 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 for acquiring the estimated rewards $\tilde{\boldsymbol{r}}^{(n)} = \{\tilde{r}_{ij}^{(n)}, i \in \mathcal{N}, j \in \mathcal{K}\}$. 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 $\tilde{\boldsymbol{r}}^{(n)}$ to yield an assignment $\boldsymbol{a}'$.
- **Exploitation phase**: this phase occupies 2^n time slots in epoch n. All users offload tasks obeying the assignment $\boldsymbol{a}'$ 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 $\boldsymbol{a}'$ to the optimal assignment $\boldsymbol{a}^*$ given an accurate reward estimation $\tilde{\boldsymbol{r}}^{(n)}$.

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.

Algorithm 1 DEBO: Decentralized Epoch Based Offloading

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

2) Exploration of RO: Users will locally execute the RO as presented in Algorithm 2. Since server j can support M_j tasks, so we regard it has M_j resource units. Accordingly, integer H_i in Line 3 entails the selected server index (resource unit), and enables task abandonment if exceeding the server capacity (Line 11). Note that $\boldsymbol{V}$ and $\boldsymbol{S}$ in Line 6 record the *successful* offloading times and accumulated rewards, respectively, which leverage all observations during the exploration to learn $\tilde{\boldsymbol{r}}^{(n)}$.

Algorithm 2 RO: Random Offloading

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

3) Matching of DAuction: The DAuction is shown in Algorithm 3. Specifically, DAuction uses the estimated rewards $\tilde{\boldsymbol{r}}^{(n)}$ to decide the assigned edge server to each user, while the optimal assignment $\boldsymbol{a}^*$ will be deduced if every learned reward $\tilde{r}_{ij}^{(n)}$ converges to the expected value μ_{ij}.

Algorithm 3 DAuction: Decentralized Auction

```
Input: $\{M_j, j \in \mathcal{K}\}$, $M$, $\tilde{\boldsymbol{r}}^{(n)}$, $T_2$, $\epsilon$
1: Initialization: $\boldsymbol{R} = \{R_{im}, \forall i \in \mathcal{N}, m = 1, ..., M\}$, set
   $\boldsymbol{B} = \{B_{im} = 0, \forall i \in \mathcal{N}, m = 1, ..., M\}$ and $\boldsymbol{a} = \boldsymbol{0}$;
2: for $i \in \mathcal{N}$ do
3:     for $m = 1, ..., M$ 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: for $T_2$ time slots do
6:     for user $i \in \mathcal{N}$ do
7:         if $a_i = 0$ then
8:             Find $m^* \in \arg\max_m \{R_{im} - B_{im}\}$;
9:             Find $m' \in \arg\max_{m \notin M(m^*)} \{R_{im} - B_{im}\}$;
10:            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;
11:    for edge server $j \in \mathcal{K}$ do
12:        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 the $m$-th unit;
13:        If the $m$-th unit is not allocated, select the highest-bid
           user to allocate the $m$-th unit;
14: for $i \in \mathcal{N}$ do
15:     $a'_i = j(a_i)$;
16: return $\boldsymbol{a}' = \{a'_i, i \in \mathcal{N}\}$;
```

Denote range $M(m) = (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^{j} M_l]$ and server index $j(m) = j$, if $m \in (\sum_{l=1}^{j-1} M_l, \sum_{l=1}^{j} M_l]$. Lines 2-4 state the initialization of rewards $\boldsymbol{R}$ for each user pertaining to M resource units. To simplify notation, assignment $a_i \in [1, M]$ means that user i will offload tasks to server $j(a_i)$ by holding the resource unit a_i (Line 10), while $a_i = 0$ (Line 7) 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. In particular, resource unit m' with the second highest value in Line 9 is attained by precluding those units belonging to the same server to expedite the auction process [18]. On edge servers' side, they will allocate their resource units (i.e., handling the offloaded tasks) to users who have the highest bids in Lines 11-13, and meanwhile discarding tasks from unassigned users. Note that the auction is decentralized as any user i only needs to locally maintain a reward vector $\{R_{im}, m = 1, ..., M\}$, a bidding vector $\{B_{im}, m = 1, ..., M\}$, and server indices a_i, a'_i.

C. Performance Analysis of DEBO

1) Main Regret Result: We now analyze the regret $\mathcal{R}(T)$ in Eq. (4). Let $\Delta_{\min} = \min_{\boldsymbol{a} \neq \boldsymbol{a}^*} \left\{ \sum_{i=1}^{N} \mu_{ia_i^*} - \sum_{i=1}^{N} \mu_{ia_i} \right\}$, 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 $\boldsymbol{a}^*$ to OAP, otherwise $\Delta_{\min}$ implies the gap between the highest and second highest compound rewards accordingly. Moreover, denote $M_{\min} = \min_{j \in \mathcal{K}} \{M_j\}$, and $\delta_{\min} = \min_{i \in \mathcal{N}} \min_{j, j' \in \mathcal{K}, j \neq j'} \{|\mu_{ij} - \mu_{ij'}|\}$ as the minimum user reward gap with both $\Delta_{\min}, \delta_{\min} > 0$.

Theorem 1. *Let* $\epsilon = \max\{\frac{\Delta_{\min}}{5N}, \frac{\delta_{\min}}{K} - \frac{3\Delta_{\min}}{4NK}\}$*,* $T_1 = \max\Big\{\Big\lceil\frac{128N^2M(\overline{r}-\underline{r})^2}{9\Delta_{\min}^2 M_{\min}}\Big\rceil, \Big\lceil\frac{81M^2}{2M_{\min}^2}\Big\rceil\Big\}$ *and* $T_2 = \lceil NM + \frac{NM\overline{r}}{\epsilon}\rceil$*. The regret* $\mathcal{R}(T)$ *of DEBO is upper bounded by*

$$\begin{aligned}\mathcal{R}(T) &\leq \Big(T_1 N\overline{r} + T_2 N\overline{r}\Big)\log_2(T+2) + 12N^2K\overline{r} \\ &= O(\log_2 T).\end{aligned} \tag{7}$$

See Appendix A in [34] for the proof. The regret bound in Eq. (7) is obtained by *bounding the error probability* P_n that $\boldsymbol{a}' \neq \boldsymbol{a}^*$ after the n-th matching, which is shown in Lemma 4. Note that our derived $\log T$ regret is *tight*, as a lower $\log T$ regret bound is deduced by regrading the user 1 as a super-user who can control all users' offloading decisions [19].

Remark: If $\Delta_{\min}$ 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 an upper bound. Also, our regret analysis is essentially unchanged when each user i can only reach a subset of servers $\mathcal{K}_i \in \mathcal{K}$.

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 [34] Appendix B.

Lemma 1. *Let* $T_1 = \max\Big\{\Big\lceil\frac{128N^2M(\overline{r}-\underline{r})^2}{9\Delta_{\min}^2 M_{\min}}\Big\rceil, \Big\lceil\frac{81M^2}{2M_{\min}^2}\Big\rceil\Big\}$*. After the* n*-th exploration, the error probability satisfies*

$$\Pr\Big(|\tilde{r}_{ij}^{(n)} - \mu_{ij}| > \frac{3\Delta_{\min}}{8N}\Big) \leq 3NKe^{-n}. \tag{8}$$

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

3) Decentralized Matching Error: Based on estimated rewards, users execute the DAuction to yield the user-server assignment $\boldsymbol{a}'$. If the compound reward induced by $\boldsymbol{a}'$ is within a gap of $\Delta_{\min}$ to the highest outcome, then $\boldsymbol{a}'$ is in fact the optimal assignment, or $\boldsymbol{a}' = \boldsymbol{a}^*$. 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], [18] or focusing on one-to-one match [11], [20]. Prior to characterizing the matching error, we need to first show that the assignment in DAuction fulfills the ϵ-complementary slackness (ϵ-CS).

Lemma 2. *Denote* $\eta_m = \max_{i\in\mathcal{N}}\{B_{im}\}$ *as the highest bid among 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* $\boldsymbol{a}$ *satisfies* ϵ*-CS, that is*

$$R_{ia_i} - \tilde{\eta}_{j(a_i)} \geq \max_m\{R_{im} - \tilde{\eta}_{j(m)}\} - \epsilon. \tag{9}$$

See Appendix C1 in [34] for the proof. ϵ-CS implies that the assignment $\boldsymbol{a}'$ attains at least near to the optimal reward.

Lemma 3. *Denote* $\boldsymbol{a}^{(n)}$ *as the optimal assignment under* $\tilde{\boldsymbol{r}}^{(n)}$*, and* $\tilde{\Delta}_{\min} = \min_{i\in\mathcal{N},j,j'\in\mathcal{K},j\neq j'}\{|\tilde{r}_{ij}^{(n)} - \tilde{r}_{ij'}^{(n)}|\}$*. If* $K\epsilon < \tilde{\Delta}_{\min}$*, DAuction ensures* $\boldsymbol{a}' = \boldsymbol{a}^{(n)}$*. Also, there holds*

$$\sum_{i=1}^{N}\tilde{r}_{ia_i'}^{(n)} \geq \sum_{i=1}^{N}\tilde{r}_{ia_i^{(n)}}^{(n)} - N\epsilon. \tag{10}$$

Furthermore, DAuction will terminate with all users ended being assigned to servers within $T_2 = \lceil NM + \frac{NM\overline{r}}{\epsilon}\rceil$ *rounds.*

See Appendix C2 in [34] for the proof. In addition to the commonly characterized $N\epsilon$ gap [11], [17], [20], our proposed DAuction further guarantees a $K\epsilon$ 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 error characterizations in Lemmas 1 and 3, we present the error probability P_n that $\boldsymbol{a}' \neq \boldsymbol{a}^*$ with the proof in [34] Appendix C3.

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

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

IV. Extension of Decentralized Offloading

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

A. Unknown Reward Gap

In previous section, we rely on the knowledge of $\Delta_{\min}$, $\delta_{\min}$ to determine the exploration and matching lengths. In some cases, this gap information may be unavailable, thereby requiring 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 the exploration length $T_1^{(n)}$ over epoch to ensure a bounded exploration error probability, meanwhile reducing the minimum increment $\epsilon^{(n)}$, or extending the matching length $T_2^{(n)}$, so as to converge to the optimal assignment. Accordingly, we denote this decentralized offloading scheme under unknown reward gaps as U-DEBO.

Theorem 2. *Let* $\epsilon^{(n)} = c_0 n^{-\vartheta}$*,* $T_1^{(n)} = \lceil c_1 n^{\vartheta}\rceil$ *and* $T_2^{(n)} = \lceil NM + \frac{NM\overline{r}}{\epsilon^{(n)}}\rceil$ *where* c_0*,* c_1 *are constants and* $\vartheta \in (0,1)$*. The regret* $\mathcal{R}(T)$ *of U-DEBO is bounded by*

$$\begin{aligned}\mathcal{R}(T) \leq &\Big\lceil c_1\log_2^{\vartheta}(T+2)\Big\rceil N\overline{r}\log_2(T+2) \\ &+ \Big\lceil MN + \log_2^{\vartheta}(T+2)MN\overline{r}/c_0\Big\rceil N\overline{r}\log_2(T+2) \\ &+ N\overline{r}(2^{n_0}-1) + 12N^2K \\ &= O\left(\log_2^{1+\vartheta} T\right),\end{aligned} \tag{11}$$

where n_0 *is a finite integer.*

Please refer to Appendix D1 in [34] 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 with $N^{(n)} \leq M$ for system stability. Note that user entering and leaving will have different impacts. For the leaving case, even all users adhere to the 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 $P_n \leq 3N^{(n)}Ke^{-n}$ when running 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, dynamic entering causes an increase in $N^{(n)}$, 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. Similarly, refer to the decentralized offloading for dynamic entering or leaving as D-DEBO, whose regret is quantified below.

Theorem 3. *Suppose the epoch n' of the last user entering satisfies $n' \leq O(\log_2 T^{\zeta}), \zeta \in (0,1)$. Let $\epsilon = \max\{\frac{\Delta_{\min}}{5N^{(n)}}, \frac{\delta_{\min}}{K} - \frac{3\Delta_{\min}}{4N^{(n)}K}\}$, $T_1 = \max\left\{\left\lceil\frac{128(N^{(n)})^2M(\overline{r}-\underline{r})^2}{9\Delta_{\min}^2 M_{\min}}\right\rceil, \left\lceil\frac{81M^2}{2M_{\min}^2}\right\rceil\right\}$ and $T_2 = \left\lceil N^{(n)}M + \frac{N^{(n)}M\overline{r}}{\epsilon}\right\rceil$. The regret $\mathcal{R}(T)$ of D-DEBO is bounded $\mathcal{R}(T) \leq O(T^{\zeta})$.*

See Appendix D2 in [34] 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.

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 proportional fairness maximization to enable a fair reward distribution among users, by changing the objective of OAP in Eq. (3) to $\max\sum_{i=1}^{N}\ln(1+\beta\mu_{ia_i}), \beta > 0$ [21], [22]. Accordingly, let $\boldsymbol{a}^f$ be the optimal assignment to the egalitarian fairness optimization, and define the fairness regret as $\mathcal{R}^f(T) = T\sum_{i=1}^{N}\ln\left(1+\beta\mu_{ia_i^f}\right) - \sum_{t=1}^{T}\sum_{i=1}^{N}\ln\left(1+\beta\mathbb{E}[r_{ia_i(t)}(t)]\right)$.

We can still employ the DEBO in Algorithm 1 by replacing the reward estimations with their logarithm fairness values. Nevertheless, this operation is built on the premise that the fairness $\ln\left(1+\beta\tilde{r}_{ij}^{(n)}\right)$ 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 $|\tilde{r}_{ij}^{(n)} - \mu_{ij}| \leq \Delta, \forall i \in \mathcal{N}, j \in \mathcal{K}$ with $\Delta < \frac{1+\beta\underline{r}}{4\beta}$, then the fairness error fulfills*

$$\left|\ln\left(1+\beta\tilde{r}_{ij}^{(n)}\right) - \ln(1+\beta\mu_{ij})\right| \leq \frac{4\beta\Delta}{3(1+\beta\underline{r})}. \tag{12}$$

See Appendix E in [34] for the proof. Based on this lemma, we adapt the DEBO to a fair decentralized offloading F-DEBO, where the input to DAuction is $\ln\left(1+\beta\tilde{r}_{ij}^{(n)}\right)$. Also, denote $\Delta_{\min}^f = \min_{\boldsymbol{a}\neq\boldsymbol{a}^f}\left\{\sum_{i=1}^{N}\ln\left(1+\beta\mu_{ia_i^f}\right) - \sum_{i=1}^{N}\ln(1+\beta\mu_{ia_i})\right\}$ and $\delta_{\min}^f = \min_{i\in\mathcal{N}}\min_{j,j'\in\mathcal{K},j\neq j'}\{|\ln(1+\beta\mu_{ij}) - \ln(1+\beta\mu_{ij'})|\}$. Theorem 4 presents the fairness regret, while the proof is omitted since it is similar to Theorem 1.

Theorem 4. *Let $\epsilon = \max\left\{\frac{\Delta_{\min}^f}{5N}, \frac{\delta_{\min}^f}{K} - \frac{3\Delta_{\min}^f}{4NK}\right\}$, $T_1 = \max\left\{\left\lceil\frac{2048N^2M(\overline{r}-\underline{r})^2\beta^2}{81(\Delta_{\min}^f)^2M_{\min}(1+\beta\underline{r})^2}\right\rceil, \left\lceil\frac{8M(\overline{r}-\underline{r})^2\beta^2}{M_{\min}(1+\beta\underline{r})^2}\right\rceil, \left\lceil\frac{81M^2}{2M_{\min}^2}\right\rceil\right\}$ and $T_2 = \left\lceil NM + \frac{NM\ln(1+\beta\overline{r})}{\epsilon}\right\rceil$. The fairness regret $\mathcal{R}^f(T)$ of F-DEBO is bounded by*

$$\begin{aligned}\mathcal{R}^f(T) &\leq T_1N\ln(1+\beta\overline{r})\log_2(T+2)\\ &\quad + T_2N\ln(1+\beta\overline{r})\log_2(T+2)\\ &\quad + 12N^2K\ln(1+\beta\overline{r})\\ &= O(\log_2 T).\end{aligned} \tag{13}$$

V. Heterogeneous Decentralized Offloading

This section explores the task offloading under the heterogeneous resource requirement, where the 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 therefore 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 $\boldsymbol{a}$ 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:

$$\begin{aligned}\max\ &\sum_{i\in\mathcal{N}_j}\mu_{ij}\\ \text{s.t. } &C_j' \leq C_j.\end{aligned} \tag{14}$$

In fact, $\alpha = 1$ implies deriving the optimal solution to Eq. (14), which is also the underlying reason why we set the benchmark for the regret definition in Eq. (6) as $\frac{1}{2}\sum_{i=1}^{N}\mu_{ia_i^*}$.

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 $\boldsymbol{I} = \{I_i, i \in \mathcal{N}\}$ 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 sub-problems. Specifically, initialize each user as unassigned $\boldsymbol{I} = \boldsymbol{0}$, and define the reward $\Delta\mu_{ij}$ to mark the performance improvement if user i's offloaded task is processed by another edge server:

$$\Delta\mu_{ij} = \begin{cases} \mu_{ij} & \text{if } I_i = 0, \\ \mu_{ij} - \mu_{ij'} & \text{if } I_i = j'. \end{cases} \tag{15}$$

According to [15], we solve Eq. (14) 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 Eq. (14) 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 $j \to j+1$. As a result, the final indicator $\boldsymbol{I}$ is guaranteed to be a 2-approximate assignment to H-OAP if one can *optimally tackle all K* subproblems. Considering the knapsack sub-problem is NP-hard, we implement a branch-and-bound method to efficiently search its optimal solution for ensuring $\alpha = 1$ [23].

B. Heterogeneous Decentralized Epoch Based 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* $\tilde{\boldsymbol{r}}^{(n)}$ 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 $\overline{M}_{\min} = \frac{\min_{j\in\mathcal{K}}\{C_j\}}{\max_{i\in\mathcal{N}}\{c_i\}}$ 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 3 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 in a *round-robin* fashion for reward exploration, where the group size is set to $\overline{M}_{\min}$ so not to violate any server capacity. Lines 5-6 are for group number $N_g \le K$ while Lines 8-10 amount for $N_g > K$. 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 $\boldsymbol{r}^{(n)}$. It should be noted that the group offloading remains decentralized because any user i can determine its group only based on the local user index i.

3) Matching and Exploitation Phases: Similar to previous analysis, the matching phase mainly entails solving the knapsack sub-problem of Eq. (14) successively. To this end, users will locally compute the rewards $\Delta\tilde{r}_{ij}^{(n)}$, which are sent to each server for updating the indicator $\boldsymbol{I}$ (i.e., offloaded task is processed) based on the branch-and-bound method (Lines 12-15). If every estimated reward $\tilde{r}_{ij}^{(n)}$ closely approaches the expected reward μ_{ij}, the obtained $\boldsymbol{a}'$ is guaranteed to be a 2-approximate assignment. Also, the matching phase lasts for K time slots corresponding to K edge servers. Finally, users will exploit the assignment $\boldsymbol{a}'$ for 2^n time slots, while other dominated exploitation length enables the same effect.

Algorithm 4 H-DEBO: Heterogeneous Decentralized Epoch Based Offloading

```
Input: ${C_j, j \in \mathcal{K}}$, ${c_i, i \in \mathcal{N}}$, $\overline{M}_{\min}$, $K$, $T_1$
 1: Initialization: Set estimated rewards $\tilde{\boldsymbol{r}}^{(n)} = \{\tilde{r}_{ij}^{(n)} = 0, \forall i \in \mathcal{N}, j \in \mathcal{K}\}$;
 2: for epoch $n = 1$ to $n_T$ do
 3:   Users form $N_g$ groups with group size being $\overline{M}_{\min}$;
        ▷ Start Exploration Phase
 4:   for $t = 1$ to $T_1$ do
 5:     if $N_g \le K$ then
 6:       Users in group $k$ offload tasks to server $[(t-1)\%K + k]\%(K+1)$, update $\tilde{r}_{ij}^{(n)}$ using $r_{ij}(t)$;
 7:     else
 8:       for $g = 1$ to $\lfloor N_g/K \rfloor$ do
 9:         Users in group $k, k = (g-1)K+1, ..., gK$ offload tasks to server $[(t-1)\%K + k - (g-1)K]\%(K+1)$, update $\tilde{r}_{ij}^{(n)}$ using $r_{ij}(t)$;
10:       Users in group $k, k = \lfloor N_g/K \rfloor K + 1, ..., N_g$ offload tasks to server $[(t-1)\%K + k - \lfloor N_g/K \rfloor K]\%(K+1)$, update $\tilde{r}_{ij}^{(n)}$ using $r_{ij}(t)$;
11:   Initialize the indicator vector $\boldsymbol{I} = \{I_i = 0, i \in \mathcal{N}\}$;
        ▷ Start Matching Phase
12:   for edge server $j \in \mathcal{K}$ do
13:     Each user $i$ computes $\Delta\tilde{r}_{ij}^{(n)}$ similar to Eq. (15), offloads a task and sends $\Delta\tilde{r}_{ij}^{(n)}$ to server $j$;
14:     Server $j$ performs branch-and-bound to solve sub-problem of Eq. (14) with input $\{\Delta\tilde{r}_{ij}^{(n)}, i \in \mathcal{N}\}$;
15:     Each user $i$ updates $I_i = j$ if assigned to server $j$;
16:   Set $\boldsymbol{a}' = \boldsymbol{I}$;                ▷ Assignment from matching
17:   for remaining $2^n$ time slots do
18:     Each user $i$ offloads tasks to the assigned server $a_i'$;
        ▷ Exploitation Phase
```

C. Performance Analysis of H-DEBO

Now we analyze the regret $\tilde{\mathcal{R}}(T)$ in Eq. (6). Let $\delta_{\min}^{(1)} = \min_{i,i'\in\mathcal{N},i\ne i'} \min_{j,k\in\mathcal{K},j\ne k} |\mu_{ij} - (\mu_{i'j} - \mu_{i'k})|$, $\delta_{\min}^{(2)} = \min_{i,i'\in\mathcal{N},i\ne i'} \min_{j,j',k\in\mathcal{K},j\ne j',j\ne k} |(\mu_{ij} - \mu_{ij'}) - (\mu_{i'j} - \mu_{i'k})|$, $\delta_{\min} = \min_{i\in\mathcal{N}} \min_{j,j'\in\mathcal{K},j\ne j'} \{|\mu_{ij} - \mu_{ij'}|\}$. Denote $\delta'_{\min} = \min\left\{\delta_{\min}^{(1)}, \delta_{\min}^{(2)}, \delta_{\min}\right\}$, $\overline{M}_{\max} = \frac{\max_{j\in\mathcal{K}}\{C_j\}}{\min_{i\in\mathcal{N}}\{c_i\}}$. The following theorem states the regret with its proof in [34] Appendix F.

Theorem 5. *Let $T_1 = \left\lceil \frac{25\overline{M}_{\max}^2(\overline{r}-\underline{r})^2}{2(\delta'_{\min})^2} K \right\rceil$ if $N_g \le K$ and $T_1 = \left\lceil \frac{25\overline{M}_{\max}^2(\overline{r}-\underline{r})^2}{2(\delta'_{\min})^2} (K+N) \right\rceil$ if $N_g > K$. The regret $\tilde{\mathcal{R}}(T)$ of H-DEBO is upper bounded by*

$$\begin{aligned} \tilde{\mathcal{R}}(T) &\le \left(T_1 \frac{N\overline{r}}{2} + K\frac{N\overline{r}}{2}\right)\log_2(T+2) + 4N^2K\overline{r} \\ &= O(\log_2 T). \end{aligned} \tag{16}$$

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 $\delta'_{\min}$ 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 our proposed decentralized offloading schemes and baseline methods.

A. Evaluation Setup

1) Parameter Setting: Consider the MEC system (divided into cells) is composed of $K = 3$ 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 4G uplink speed [24]. Server capacity for maximum task service M_j is an integer in $[2, 5]$, while for endowed computing resource C_j is randomly in $[2, 3.5]$ with user resource requirement $c_i \in [0.5, 1]$. The reward preference function of Eq. (2) 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 $\underline{r} = 0.3, \overline{r} = 3.8$ 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.

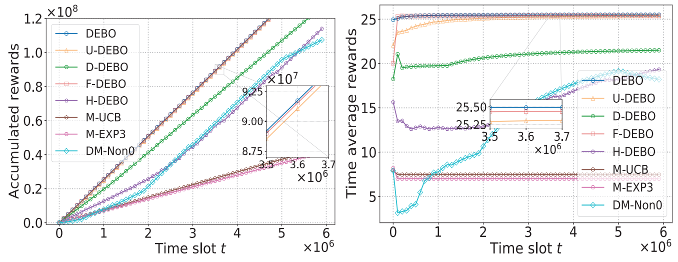
- M-UCB: upper confidence bound (UCB) algorithm which pulls arm with the highest confidence bound of the average reward [25]. Considering that there are multiple users, suppose they independently execute UCB for task offloading, namely M-UCB.
- M-EXP3: exponential-weight algorithm for exploration and exploitation (EXP3) that pulls arms following the exponential-weight probability [26]. For our problem, multiple users will locally choose edge servers based on EXP3, i.e., M-EXP3.
- DM-Non0: decentralized multi-user MAB with non-zero rewards [27]. 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.

The optimal assignment $\boldsymbol{a}^*$ to OAP is obtained through Hungarian algorithm [28] by splitting each server j into M_j copies, and is utilized to compute the regret $\mathcal{R}(T)$ in Eq. (4).

B. Evaluation Results over Time

We first show the performance over time given $N = 10$.

Accumulated/average rewards. The accumulated and time average rewards of our proposed and baseline algorithms are demonstrated in Fig. 2. One can observe that our proposed algorithms outperform the benchmarks as they 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 and F-DEBO also have satisfactory performances with comparable rewards to DEBO. Besides, D-DEBO leads to slightly lower rewards mainly because dynamic user leaving will cause a reduction in the compound reward $\sum_{i=1}^{N^{(n)}} \mu_{ia_i}$. The average



(a) Accumulated rewards (b) Average rewards

Fig. 2: Offloading rewards over time.

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. The baseline method DM-Non0 acquires higher rewards than M-UCB and M-EXP3, i.e., strawman extension of single-user UCB and EXP3 to multi-user scenarios is insufficient to obtain satisfactory performances.

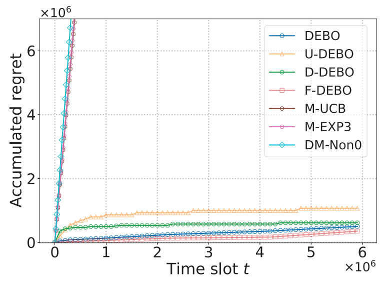
Accumulated/average regret and ratio. The accumulated and time average regrets are shown in Fig. 3(a)-(b), where H-DEBO is excluded as we can not derive the optimal assignment to H-OAP. Note that our proposed algorithms give rise to orderly lower regrets than the benchmarks. The regrets of DEBO, U-DEBO and F-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. We then display the ratio between time average rewards and the optimal result in Fig. 3(c), which indicates 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 M-UCB, M-EXP3 and DM-Non0, there exist performance gaps between their ratios and 1.

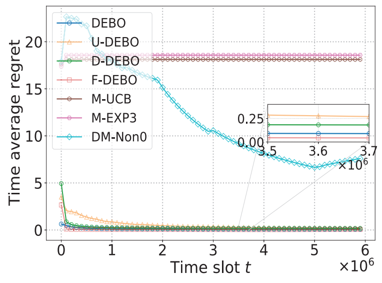
C. Evaluation Results over User Number

Varying the number of users N from 8 to 12, 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. 4-6, respectively. With the increase of N, the rewards of all algorithms will tend to increase. Besides, our proposed DEBO, U-DEBO, D-DEBO, F-DEBO and H-DEBO always achieve better performances than baseline methods M-UCB, M-EXP3 and DM-Non0 pertaining to rewards, regret and ratio.

Fairness demonstration. F-DEBO is proposed to ensure a more equitable reward distribution among users by introducing the proportional fairness. Previously, we have already shown that F-DEBO has comparable rewards to DEBO. Fig. 7 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 particular, F-DEBO accomplishes 51.49% fairness improvement with only sacrificing 0.44% compound reward on average.


(a) Accumulated regret


(b) Average regret

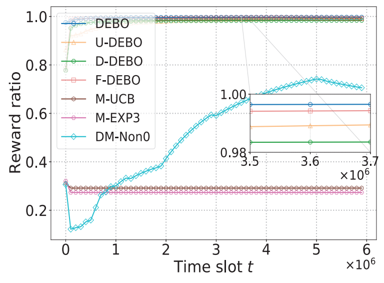

(c) Reward ratio

Fig. 3: Offloading regret and reward ratio over time.

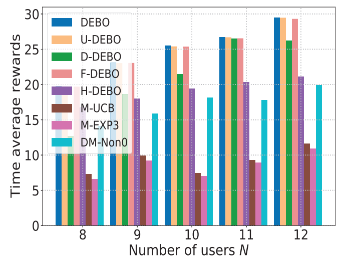


Fig. 4: Rewards vs. N.

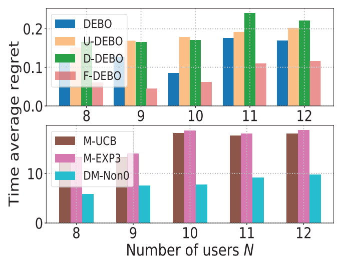


Fig. 5: Regret vs. N.

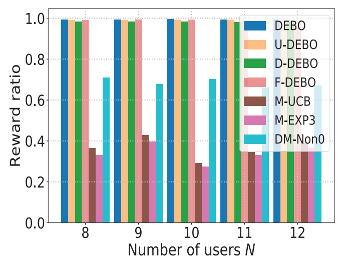


Fig. 6: Reward ratio vs. N.

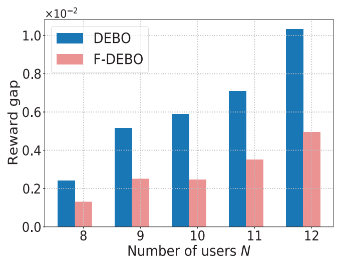


Fig. 7: Reward gap vs. N.

VII. Related Work

In this paper, we study the decentralized task offloading in a dynamic MEC system with unknown system-side information. In the following, 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 [29]. 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 [30]. Similarly, the authors in [31] also 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 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 [27] 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 problem 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 supposed to be higher than a predefined proportion [32], or multi-user reward fairness through centralized decision making [33]. A collision based multi-user reward fairness is also explored in [19].

VIII. Conclusion

In this paper, we study a fully decentralized task offloading for a dynamic MEC system with the unknown system-side information. We divide the time horizon into epochs and propose DEBO which ensures an $O(\log T)$ regret of the dynamic task offloading in a decentralized manner. On this basis, we also extend DEBO to handle cases such as the unknown reward gap, dynamic user entering or leaving, and fair reward distribution, where sub-linear regrets are obtained for all extensions. Considering users' heterogeneous resource requests, we further develop H-DEBO which achieves an $O(\log T)$ 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," Available: https://campustechnology.com/articles/2015/04/15/internet-of-things-data-to-top-1-6-zettabytes-by-2020.aspx

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE IoT*, vol. 3, no. 5, pp. 637-646, 2016.

[4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM TON*, vol. 24, no. 5, pp. 2795-2808, 2015.

[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 ICNP*, 2021, pp. 1-11.

[6] S. Sundar, and B. Liang, "Offloading Dependent Tasks with Communication Delay and Deadline Constraint," in *Proc. IEEE INFOCOM*, 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 JSAC*, vol. 35, no. 1, pp. 2637-2646, 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 INFOCOM*, 2019, pp. 1468-1476.

[9] L. Chen, and J. Xu, "Task Replication for Vehicular Cloud: Contextual Combinatorial Bandit with Delayed Feedback," in *Proc. IEEE INFOCOM*, 2019, pp. 748-756.

[10] I. Bistritz, and A. Leshem, "Distributed Multi-Player Bandits-a Game of Thrones Approach," in *Proc. NeurIPS*, 2018.

[11] S. M. Zafaruddin, I. Bistritz, A. Leshem, and D. Niyato, "Distributed Learning for Channel Allocation Over a Shared Spectrum," [Online]. Available: https://arxiv.org/pdf/1902.06353.pdf

[12] S. J. Darak, and M. K. Hanawal, "Multi-Player Multi-Armed Bandits for Stable Allocation in Heterogeneous Ad-Hoc Networks," *IEEE JSAC*, vol. 37, no. 10, pp. 2350-2363, 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 JSAC*, vol. 33, no. 12, pp. 2510-2523, 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. IEEE ICT*, 2016, pp. 1-5.

[15] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the Generalized Assignment Problem," *Information Processing Letters*, 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 INFOCOM*, 2019, pp. 1693-1701.

[17] N. Nayyar, D. Kalathil, and R. Jain, "On Regret-Optimal Learning in Decentralized Multiplayer Multiarmed Bandits," *IEEE TCNS*, vol. 5, no. 1, pp. 597-606, 2018.

[18] D. P. Bertsekas, and D. A. Castanon, "The auction algorithm for the transportation problem," *Annals of Operations Research*, vol. 20, no. 1, pp. 67–96, 1989.

[19] 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. ICML*, 2020, pp. 930-940.

[20] O. Naparstek, and A. Leshem, "Fully Distributed Optimal Channel Assignment for Open Spectrum Access," *IEEE TSP*, vol. 62, no. 2, pp. 283-294, 2014.

[21] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33-37, 1997.

[22] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *Proc. IEEE INFOCOM*, 2018, pp. 2420-2428.

[23] R. E. Neapolitan, and K. Naimipour, *Foundations of Algorithms Using C++ Pseudocode*, Jones and Bartlett Publishers, 2004.

[24] 4G4U, "4G Speed Tests," Available: https://www.4g4u.org/4g-speed-tests/

[25] P. Auer, N. C. Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.

[26] 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.

[27] A. Magesh, and V. V. Veeravalli, "Decentralized Heterogeneous Multi-Player Multi-Armed Bandits with Non-Zero Rewards on Collisions," [Online]. Available: https://arxiv.org/pdf/1910.09089.pdf

[28] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83-97, 1955.

[29] 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. ACM ICPP*, 2020, pp. 1-11.

[30] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE TMC*, vol. 17, no. 8, pp. 1791–1805, 2018.

[31] 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 JSAC*, vol. 36, no. 9, pp. 2114–2127, 2018.

[32] F. Li, J. Liu, and B. Ji, "Combinatorial Sleeping Bandits With Fairness Constraints," in *Proc. IEEE INFOCOM*, 2019, pp. 1702-1710.

[33] S. Hossain, E. Micha, and N. Shah, "Fair Algorithms for Multi-Agent Multi-Armed Bandits," [Online]. Available: https://arxiv.org/pdf/2007.06699.pdf

[34] X. Wang, J. Ye, and J. C.S. Lui, "Decentralized Task Offloading in Edge Computing: A Multi-User Multi-Armed Bandit Approach," [Online]. Avaliable: https://arxiv.org/abs/2112.11818