Bernd Simon, Helena Mehler and Anja Klein, ”Online Learning in Matching Games for Task Offloading in Multi-Access Edge Computing,” in *Proc. of the IEEE International Conference on Communications - (IEEE ICC 2023)*, May 2023.

Online Learning in Matching Games for Task Offloading in Multi-Access Edge Computing

Bernd Simon, Helena Mehler, Anja Klein
Communications Engineering Lab, TU Darmstadt, Germany, {b.simon, a.klein}@nt.tu-darmstadt.de

***Abstract*—In multi-access edge computing (MEC), mobile users (MUs) can offload computation tasks to nearby computational resources, which are owned by a mobile network operator (MNO), to save energy. In this work, we investigate two important challenges of task offloading in MEC: (i) The techno-economic interactions of the MNO and the MUs. The MNO faces a profit maximization problem, whereas the MUs face an energy minimization problem. (ii) Limited information at the MUs about the MNO's communication and computation resources and the task offloading strategies of other MUs. To overcome these challenges, we model the task offloading problem as a matching game between the MUs and the MNO including their techno-economic interactions. Furthermore, we propose a novel Collision-Avoidance Task Offloading Multi-Armed-Bandit (CA-TO-MAB) algorithm, that allows the MUs to learn the amount of available resources at the MNO and the task offloading strategies of other MUs in an online, fully decentralized way. We show that by using CA-TO-MAB, the cumulative revenue of the MNO can be increased by 25% and, at the same time the energy consumption of the MUs can be reduced by 6% compared to state-of-the-art online learning algorithms for task offloading. Furthermore, the communication overhead can be reduced by 55% compared to a non-learning game-theoretic approach.**

I. Introduction

Multi-access edge computing (MEC) refers to the deployment of computation facilities at the edge of the Internet, aiming to reduce computation time and energy consumption of battery-powered mobile units (MUs) [1]. The mobile network operators (MNOs) deploy cloudlets, i.e., small-scale data centers, at the wireless access points (APs). MEC enables computation offloading, which is the transmission of computation tasks from MUs to nearby APs with cloudlets, where the task can be efficiently computed. From the perspective of battery-powered MUs, computation offloading offers the possibility to reduce the energy consumption caused by intensive computation tasks, therefore extending their operation time. Recent works have proposed using computation offloading in face recognition [2] and video analysis [3].

In MEC, the limited communication bandwidth and computation resources have to be shared among many MUs. After successfully offloading computation tasks, the MUs pay the MNO to compensate for the resource usage [4]. This techno-economic perspective, i.e., considering the economic attributes as well as the technical performance of MEC, is becoming increasingly important.

This work has been funded by the German Research Foundation (DFG) as a part of the projects B3, C1 and T2 within the Collaborative Research Center (CRC) 1053 - MAKI (Nr. 210487104) and has been supported by the BMBF project Open6GHub (Nr. 16KISK014) and the Loewe Center EmergenCity.

The success of MEC relies on suitable task offloading strategies that minimize the MU's energy consumption and maximize the MNO's revenue. However, designing the task offloading strategy in MEC networks faces two major challenges:

(i) The techno-economic interactions between MUs and the MNO: Both, the MU and the MNO have technical and economic perspectives. The MUs want to minimize their energy consumption given a fixed willingness to pay for saved energy. The MNO maximizes its revenue given the available limited communication and computation resources.

(ii) Limited information: The MUs do not know how many resources are available at each AP and what the task offloading strategies of the other MUs are. To avoid collisions, i.e. more MUs propose to offload their task to an AP than it can process, the MUs have to learn how to coordinate their task offloading strategies in a decentralized way.

Recent research on task offloading in MEC has been focused on either the techno-economic interactions or the decentralized online learning of the task offloading strategy. Centralized decision-making, where tasks are scheduled based on global information and MUs have no control, will not be considered as it is not feasible due to the significant communication overhead in dense MEC environments [5]. In [6], a techno-economic model for a MEC network is formulated based on matching theory. This game is solved using a swap-matching algorithm. A techno-economic model for a MEC system with a controller is formulated in [5]. A reinforcement learning algorithm for the MUs' offloading strategy is proposed which relies on a central controller. The central controller minimizes the time occupation of the edge servers. Although these works significantly contribute to the research on techno-economic interactions for task offloading, these works either assume that the MUs have unlimited information or these works rely on a central controller in the MEC network.

Several works considered the decentralized online learning of the MUs' task offloading strategies. In [7], a decentralized multi-user multi-armed bandit is proposed to learn the offloading decisions. In [8], a delay minimization problem is investigated with a single AP and multiple MUs using reinforcement learning. A multi-armed bandit learning that minimizes the average task completion time of the MUs was proposed in [9]. A distributed deep learning-based offloading algorithm for a hierarchical MEC network with multiple edge servers and one cloud server that minimizes a combination of task completion time and energy consumption was proposed

in [10]. [11] formulates a utility model for MUs and proposes a model-free reinforcement learning offloading mechanism so that MUs learn long-term offloading strategies. One approach to edge server behavior is proposed in [12] where the edge servers determine their activity status, while MUs aim to attain Quality of Service satisfaction by choosing active servers and determining transmission powers. Although these works significantly contribute to the research of decentralized online learning of task offloading strategies, these works only consider the perspective of the MUs, without modeling the behavior of the MNO.

Our contributions are summarized as follows. In contrast to [7], [8], [9], [10], [11], we present a techno-economic system model consisting of two stakeholder types, the MUs and the MNO. We present a matching game formulation of the task offloading problem including the techno-economic perspective. We consider the relevant utility functions, i.e., the energy consumption of each MU and the revenue of the MNO in the game formulation. To find solutions to the formulated matching game in a completely decentralized way, we combine elements from online learning, i.e., multi-armed bandits, with concepts from game theory, i.e., coordination of the MUs' offloading strategies. Furthermore, in contrast to [6], [5], we propose a fully decentralized online learning of the task offloading strategies termed Collision-Avoidance Task Offloading Multi-Armed-Bandit (CA-TO-MAB).

The rest of this paper is organized as follows. In Section II, we introduce the system model, followed by the formulation of the task offloading game in Section III. The novel CA-TO-MAB algorithm is presented in Section IV and the performance of the proposed approach is evaluated numerically in Section V. Section VI concludes this paper.

II. System Model

A. Overview

We consider a set $\mathcal{K} = \{\mathrm{MU}_1, ..., \mathrm{MU}_K\}$ of K MUs. One MNO provides a set $\mathcal{M} = \{\mathrm{AP}_1, ..., \mathrm{AP}_M\}$ of M access points (APs) with one cloudlet at each AP. An overview of the system model is depicted in Fig. 1.

The time is divided into discrete time slots with index $i = i, \dots, I$. In each time slot i, a computation task arrives at each of the K MUs. Each task can be processed locally on the MU or can be offloaded to one of the cloudlets in the MEC network using the shared wireless channel. The task is delay-sensitive, meaning that in the case of task offloading, the task completion time for task offloading needs to be the same or smaller than the local computation time. When offloading the task to a cloudlet in the MEC network, the MU sends an offloading proposal to the selected cloudlet including information about the task and a suggested payment. The MNO then decides whether the offloading proposal will be accepted or if the task has to be calculated locally.

B. Mobile Units

In each time slot i, a delay-sensitive task arrives at MU_k, which is characterized by its size $s_{k,i}$ in bits and its complexity

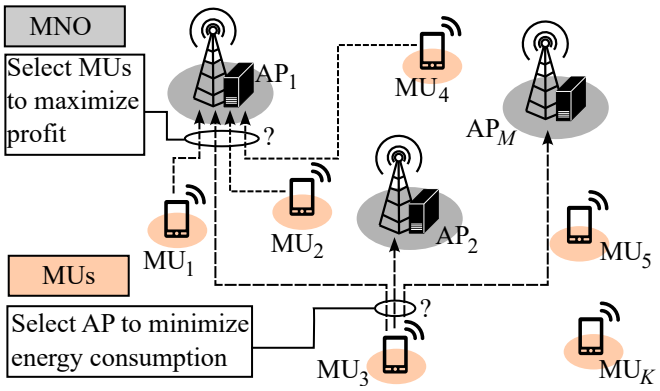

Fig. 1. Overview of the system model.

$c_{k,i}$ in CPU cycles per bit [13]. There are two different ways the task can be completed: Local computation at the MU or task offloading to a cloudlet in the MEC network. When calculating the task locally, the MU uses its central processing unit (CPU) with the frequency f_k^{local} and the power p_k^{calc}. For local computation of the task in time slot i at MU k, the processing time is calculated by

$$T_{k,i}^{\text{local}} = \frac{s_{k,i} \cdot c_{k,i}}{f_k^{\text{local}}}. \tag{1}$$

The energy required for local task completion in i is given by

$$E_{k,i}^{\text{local}} = T_{k,i}^{\text{local}} \cdot (p_k^{\text{static}} + p_k^{\text{calc}}), \tag{2}$$

where p_k^{static} denotes the static power of MU_k.

Alternatively, the task can be offloaded to a cloudlet. We introduce the indicator variable $x_{k,m,i} \in \{0, 1\}$ to indicate whether MU_k offloads its task to AP_m ($x_{k,m,i} = 1$) or calculates the task locally ($x_{k,m,i} = 0, \forall m$). The task can only be offloaded to one AP, i.e., $x_{k,m,i} = 1$ for only one $m \in \{1, \dots, M\}$. We introduce the task offloading matrix $\mathbf{X}_i = (x_{k,m,i})$ which contains the information about which MU offloads its task to which AP in time slot i. When offloading the task, the task has to be transmitted over the shared wireless channel. We assume a block fading model, i.e. the channel is assumed to be constant during time slot i, but changes in between time slots. The gain of the wireless channel between MU k and AP m in time slot i is denoted by $|h_{k,m,i}|^2$. We assume a white Gaussian noise at the receiver with noise power σ^2. We assume Frequency Division Multiple Access (FDMA), i.e., a bandwidth $b_{k,m,i}$ is assigned to each MU. We can calculate the data rate

$$R_{k,m,i} = b_{k,m,i} \cdot \log_2 \left(1 + \frac{p_k^{\text{trans}} \cdot |h_{k,m,i}|^2}{\sigma^2}\right), \tag{3}$$

where $b_{k,m,i}$ denotes the bandwidth assigned to MU_k at AP_m in i and p_k^{trans} denotes the transmit power of MU_k. The time required for the transmission of the task from MU_k to AP_m in time slot i is given by

$$T_{k,m,i}^{\text{trans}} = \frac{s_{k,i}}{R_{k,m,i}}. \tag{4}$$

The time required for processing the task on cloudlet m is given by

$$T^{\text{calc}}_{k,m,i} = \frac{s_{k,i} \cdot c_{k,i}}{f_{k,m,i}}, \tag{5}$$

where $f_{k,m,i}$ denotes the CPU frequency assigned to MU k on cloudlet m in i. The total time required for task completion is given by $T^{\text{offloading}}_{k,m,i} = T^{\text{trans}}_{k,m,i} + T^{\text{calc}}_{k,m,i}$. As the tasks are delay-sensitive, the constraint

$$T^{\text{offloading}}_{k,m,i} \leq T^{\text{local}}_{k,i} \tag{6}$$

is important. The total energy required at MU_k for its task completion on cloudlet m is given by

$$E^{\text{offloading}}_{k,m,i} = T^{\text{trans}}_{k,m,i} \cdot p^{\text{trans}}_k + T^{\text{calc}}_{k,m,i} \cdot p^{\text{static}}_k, \tag{7}$$

which includes the energy required for the transmission of the task. The utility of MU_k in time slot i

$$U^{\text{MU}}_{k,m,i} = \begin{cases} E^{\text{local}}_{k,i} - E^{\text{offloading}}_{k,m,i}, & \text{if } x_{k,m,i} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

is the energy saved by the task offloading to AP_m. Each MU has a different willingness to pay the MNO for the successful task offloading which is modeled by the factor α_k [14], [15]. MU_k pays the MNO for successful task offloading depending on the saved energy with a payment

$$P_{k,m,i} = \begin{cases} \alpha_k \cdot U^{\text{MU}}_{k,m,i}, & \text{if } T^{\text{offloading}}_{k,m,i} \leq T^{\text{local}}_{k,i} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

in time slot i. If the task completion time constraint (6) is violated, the MU will not pay for the task offloading.

Each MU solves an *energy minimization problem*, i.e. selecting a cloudlet to maximize the saved energy (8) while keeping the task completion time constraint (6) of each task.

C. Mobile Network Operator

The MNO owns and operates the APs and cloudlets. Each AP has a limited bandwidth $B^{\text{max}}_{m,i}$ of the wireless channel, which has to be shared between the offloading MUs. Furthermore, each cloudlet has a limited CPU frequency $F^{\text{max}}_{m,i}$, which has to be shared between the offloaded tasks. Therefore, the MNO has to fulfill the constraints

$$\sum_{k=1}^{K} b_{k,m,i} \leq B^{\text{max}}_{m,i} \quad \text{and} \quad \sum_{k=1}^{K} f_{k,m,i} \leq F^{\text{max}}_{m,i} \quad \forall m, i. \tag{10}$$

Given these constraints, the MNO wants to maximize its utility

$$U^{\text{MNO}}_i(\mathbf{X}_i) = \sum_{m=1}^{M} \sum_{k=1}^{K} x_{k,m,i} P_{k,m,i} \tag{11}$$

in time slot i given by the sum of all payments by the MUs.

The MNO solves a *profit maximization problem*, i.e. selecting the MUs to maximize its profit (11) given the constraint of the available communication and computation resources (10).

III. Task Offloading Game

A. Game formulation

We use game theory, specifically matching theory, to analyze the task offloading problem. The main goal of matching theory is to obtain a stable matching, i.e., reaching a task offloading matrix $\mathbf{X}_i$ in which MUs and the MNO cannot simultaneously improve their utility by changing their offloading decisions. This reflects the selfish behavior of MUs and the MNO that try to obtain their individual best solution.

A matching game is characterized by the set of players and their preferences. The MUs are modeled as individual players. The preferences of the MUs are given by the preference ranking

$$\text{AP}_l \succeq^{\text{MU}}_k \text{AP}_n \iff U^{\text{MU}}_{k,l,i} \geq U^{\text{MU}}_{k,n,i}, \tag{12}$$

i.e., MU_k prefers AP_l over AP_n if the MU's utility is higher when using AP_l. The MUs rank all APs in the set $\mathcal{M}$ of available APs according to their utility. This ranking is different for each MU.

The MNO is modeled as a player which has preferences regarding the offloading MUs at each AP. The MNO ranks sets of MUs at each AP

$$C \succeq^{\text{AP}}_m C' \iff U^{\text{MNO}}_i(\mathbf{X}_i) \geq U^{\text{MNO}}_i(\mathbf{X}'_i), \tag{13}$$

i.e., AP_m prefers the set C of MUs over the set C' if the MNO's utility is higher. The task assignment matrix $\mathbf{X}_i$ is a matrix indicating that the set C of MUs is offloading to AP_m, $\mathbf{X}'_i$ is the matrix indicating that C' is offloading to AP_m.

The task offloading game $\mathcal{G}_i$ in time slot i is formally described by a tuple $\mathcal{G}_i = (\mathcal{K}, \mathcal{M}, \succeq^{\text{MU}}_k, \succeq^{\text{AP}}_m)$ containing the set $\mathcal{K}$ of MUs, the set $\mathcal{M}$ of APs, the MUs' preference ordering $\succeq^{\text{MU}}_k$ and the APs' preference ordering $\succeq^{\text{AP}}_m$. We denote the set of MUs that are matched to AP_m as $\mu(\text{AP}_m)$, and the AP to which the MU is matched to as $\mu(\text{MU}_k)$.

B. Stable solution

Comparable to the concept of the Nash Equilibrium (NE), a solution of the matching game $\mathcal{G}_i$ is a task offloading matrix $\mathbf{X}_i$ where neither the MNO nor the MUs can simultaneously improve their utility by changing their strategy. In matching games, both, the MUs and the MNO have to agree to change the matching, therefore the solution concept is different from the NE.

A matching is defined to be stable when there is no pair of an AP_m and a set C of MUs that can further improve their utility by changing the current matching. A pair (AP_m, C) is called a blocking pair if the AP_m and all MUs in C can improve their utility [16]. Formally, the pair (AP_m, C) is a blocking pair if all of the following three conditions are fulfilled:

(i) $C \setminus \mu(\text{AP}_m) \neq \{\}$, i.e., not all MUs C are currently matched to AP_m.

(ii) $C \succeq^{\text{AP}}_m \mu(\text{AP}_m)$, i.e., the MNO prefers to have C matched to the AP_m over its current matching $\mu(\text{AP}_m)$.

(iii) $\text{AP}_m \succeq^{\text{MU}}_k \mu(\text{MU}_k), \forall \text{MU}_k \in C$, i.e., all MUs in C

prefer to be matched to AP_m over their current matching $\mu(\mathrm{MU}_k)$.

If one blocking pair (AP_m, C) exists, the matching is defined to be unstable as it can be improved by offloading all the tasks of the MUs in C to AP_m. Conversely, if no blocking pair exists, the matching is defined to be stable.

C. Available information

Initially, the MUs have no information about the maximum available communication bandwidth $B_{m,i}^{\max}$ and computation resources $F_{m,i}^{\max}$ at each AP_m. Furthermore, the MUs do not know how much bandwidth $b_{k,m,i}$ and CPU frequency $f_{k,m,i}$ they get assigned at each AP_m. This depends heavily on the offloading strategies of the other MUs and the decisions of the MNO. We define the expected allocated bandwidth as $\bar{b}_{k,m} = \mathbb{E}\{b_{k,m,i}\}$ and the expected allocated CPU frequency as $\bar{f}_{k,m} = \mathbb{E}\{f_{k,m,i}\}$. We define the MU-side information as $\mathcal{I}_k^{\mathrm{MU}} = \{\bar{b}_{k,m}, \bar{f}_{k,m}, \forall \mathrm{AP}_m \in \mathcal{M}\}$, which is the information about the expected bandwidth and expected CPU frequency that will be assigned to MU_k at AP_m.

The MNO does not have information about the MUs in the network. In the case of an offloading proposal, the MU reports the task size $s_{k,i}$, the task complexity $c_{k,i}$, the time constraint $T_{k,i}^{\mathrm{local}}$, the channel gain $|h_{k,m,i}|^2$ and its willingness to pay α_k to the MNO.

Our goal is to optimize the task offloading matrix $\mathbf{X}_i$ in a completely decentralized fashion without requiring prior knowledge of $\mathcal{I}_k^{\mathrm{MU}}$. For this purpose, each MU learns $\mathcal{I}_k^{\mathrm{MU}}$ to find its most preferred AP in each time slot i. In turn, the MNO has to identify the most profitable MUs for each AP. We assume strict privacy constraints, meaning that the MUs do not share information about $\mathcal{I}_k^{\mathrm{MU}}$, neither with the MNO nor other MUs. Because of the limited information and privacy constraints, an efficient task assignment can only be achieved by means of decentralized online learning. A major challenge for the MUs is the exploration of the APs at the beginning, when all MUs explore simultaneously the different APs, having only poor estimates of $\mathcal{I}_k^{\mathrm{MU}}$. To analyze the task offloading problem from the perspective of the MUs and the MNO, we propose a novel decentralized learning algorithm in the next section.

IV. Collision-Avoidance Task Offloading Multi-Armed-Bandit

In this section, we propose a novel decentralized online learning algorithm for the task offloading game $\mathcal{G}_i$. The algorithm is fully decentralized and it consists of two strategies: The strategy of the MUs (Algorithm 1) and the strategy of the MNO (Algorithm 2). The collision concept of Collision-Avoidance Task Offloading Multi-Armed-Bandit is based on the collision-avoidance multi-armed bandit from [17].

The fundamental idea of our proposed CA-TO-MAB algorithm is that the MUs learn the expected utility of each AP and whether other MUs with higher willingness α_k to pay propose to the same AP. Each MU wants to avoid collisions with MUs having a higher willingness α_k to pay, as the MNO will prefer MUs with a higher α_k. Therefore, the collision-avoidance concept was proposed in [17]. CA-TO-MAB learns the set $L_{k,m,i}$ which contains all other MUs that were preferred over MU_k by the MNO at AP_m. The strategy of the MNO is to select the MUs that maximize its revenue at each AP once all MUs have sent their offloading proposals.

Algorithm 1 CA-TO-MAB (MUs' strategy)

Require: Set $\mathcal{M}$ of APs, $\lambda \in [0, 1)$, $\{\epsilon_i\}_{i=0,\dots,I}$
1: **for each** $i \in \{0, \dots, I\}$ **do**
2: **if** $i = 0$ **then**
3: Propose to a random $\mathrm{AP}_m \in \mathcal{M}$, Initialize $L_{k,m,0} = \{\}$
4: **else**
5: Draw $d \in \{0, 1\}$ from Bernoulli distribution with p = λ
6: **if** $d = 0$ **then**
7: Determine plausible set $S_{k,i}$ from $L_{k,m,i}$
8: Sample a number e uniformly from [0, 1)
9: **if** $e < 1 - \epsilon_i$ **then**
10: Send offloading proposal to $\mathrm{AP}_m \in S_{k,i}$ with highest average utility (ties broken arbitrarily)
11: **else**
12: Send offloading proposal to a random $\mathrm{AP}_m \in S_{k,i}$
13: **end if**
14: **else**
15: Propose to the same AP_m as previous time slot
16: **end if**
17: **end if**
18: The MNO uses Algorithm 2 to determine the accepted MUs
19: **if** MU_k is accepted at AP_m, i.e., $\mathrm{MU}_k \in G_m$ **then**
20: Offload the task to AP_m, $N_m \leftarrow N_m + 1$
21: Observe utility $U_{k,m,i}^{\mathrm{MU}}$ (8) and update $Q_{k,m,i}$ ▷ Eq. (14)
22: **else**
23: Calculate the task locally
24: Update $L_{k,m,i}$ with G_m ▷ Eq. (15)
25: **end if**
26: **end for**

Algorithm 1 describes the online learning process of each MU_k. In the first time slot $i = 0$, each MU initializes its set $L_{k,m,0}$ and sends its offloading proposal to a random AP (line 3). For time slots $i > 0$, each MU draws a random number $d \in \{0, 1\}$ from a Bernoulli distribution with the probability λ (line 5). Using the set $L_{k,m,i}$, the MU calculates the set $S_{k,i}$. If $d = 0$, the MU selects an AP from $S_{k,i}$ for the offloading proposal according to a decaying ϵ-greedy scheme, i.e., with probability ϵ_i the MU explores and selects a random AP (line 12) and with probability $1 - \epsilon_i$ the MU exploits its learned information and selects the AP with the highest expected average utility (line 10).

If $d = 1$, the MU sends an offloading proposal to the same AP as in the last time slot (line 16). This mechanism is required to ensure that not all MUs switch simultaneously the AP they propose to in each round, which is required to achieve stability [17].

Afterward, MU_k waits for the response of the MNO, as described in Algorithm 2. After MU_k receives the MNO's response, its behavior depends on whether it was accepted or not. If MU_k was accepted, the computation task is offloaded (line 20) and the MU can observe its utility $U_{k,m,i}^{\mathrm{MU}}$ and update

Algorithm 2 CA-TO-MAB (MNO's strategy)

Require: $\mathcal{M}$, offloading proposals σ_m

1: **for** $i = 0, \ldots, I$ **do**
2: Wait for the set σ_m of MUs' sending offloading proposals.
3: **for each** $\mathrm{AP}_m \in \mathcal{M}$ **do**
4: $G_m = \{\}$,$A = \{\}$
5: Calculate utility U_i^{MNO} (11) for each $\mathrm{MU}_k \in \sigma_m$.
6: **for** $l = 1, \ldots, |\sigma_m|$ **do**
7: Determine MU_k with highest utility.
8: **if** $U_i^{\mathrm{MNO}}(G_m \cup \mathrm{MU}_k) \leq U_i^{\mathrm{MNO}}(G)$ **then**
9: **for each** $\mathrm{MU}_g \in G$ **do**
10: **if** $U_i^{\mathrm{MNO}}((G_m \cup \mathrm{MU}_k) \setminus (A \cup \mathrm{MU}_g)) > U_i^{\mathrm{MNO}}(G_m)$ **then**
11: $A \leftarrow A \cup \mathrm{MU}_g$
12: **end if**
13: **end for**
14: **end if**
15: **if** $U_i^{\mathrm{MNO}}((G_m \cup \mathrm{MU}_k) \setminus A) > U_i^{\mathrm{MNO}}(G_m)$ **then**
16: $G_m \leftarrow ((G_m \cup \mathrm{MU}_k) \setminus A)$
17: **end if**
18: **end for**
19: Broadcast set G_m of MUs that are accepted at AP_m
20: Reserve the CPU frequency $f_{k,m,i}$ and the bandwidth $b_{k,m,i}$ for each MU_k in G_m for the time slot i ▷ Eq. (16)
21: **end for**
22: **end for**

its estimate

$$Q_{k,m,i} = Q_{k,m,i-1} + \frac{1}{N_m}(U_{k,m,i}^{\mathrm{MU}} - Q_{k,m,i-1}), \qquad (14)$$

of the expected utility, where N_m denotes how often MU_k has successfully offloaded its task to AP_m (line 21).

If MU_k was rejected by the MNO, it calculates its task locally (line 23) and therefore cannot observe the utility at AP_m. However, the MU can learn about the MUs that were more preferred by the MNO. The set $L_{k,m,i}$ is updated using

$$L_{k,m,i} = L_{k,m,i-1} \cup G_m, \qquad (15)$$

i.e., all MUs which were accepted at AP_m instead of MU_k are included in $L_{k,m,i}$ (line 24).

Algorithm 2 describes the decision-making process of the MNO for each AP_m. The selection of MUs is based on a greedy approximation of the maximum of U_i^{MNO} of (11) based on [18]. Firstly, the MNO waits for all offloading proposals by the MUs. The proposing MUs to AP_m are denoted by the set σ_m. Then at each AP_m, the set of accepted MUs and the set of rejected MUs is determined to maximize U_i^{MNO} of (11). Firstly, the marginal contribution to the MNO's utility is calculated (line 5). Secondly, the MU with the highest marginal contribution to the utility is selected (line 7). If sufficient communication and computation resources are available at AP_m (line 15), the MU is added to the set G_m of accepted MUs (line 16). We assume the MNO shares the resources according to

$$b_{k,m,i} = \frac{B_{m,i}^{\max}}{|G_m|} \text{ and } f_{k,m,i} = \frac{F_{m,i}^{\max}}{|G_m|} \quad \forall k \in G_m, m, i, \quad (16)$$

i.e., each accepted MU gets the same amount of bandwidth $b_{k,m,i}$ and CPU frequency $f_{k,m,i}$ allocated in time slot i at AP_m. When the communication and computation resources are not sufficient to compute the task within the time constraint (line 8), the MNO checks if it is beneficial to swap the proposing MU with one of the MUs in the set G_m (lines 9-12).

Then the MNO broadcasts the set G_m of accepted MUs at AP_m to all MUs that sent an offloading proposal. Furthermore, the MNO reserves the communication and computation resources for each MU in time slot i. The MUs in G_m then can offload their computation task to AP_m.

TABLE I
EVALUATION PARAMETERS

Parameter	Value
Simulated area	$1\,\mathrm{km}^2$
Available bandwidth at each AP	$B_{m,i}^{\max} = 2\,\mathrm{MHz}$
Available CPU frequency at each AP	$F_{m,i}^{\max} \sim \mathcal{N}(30, 5)\mathrm{GHz}$
Channel gain in distance $d_{k,m}$	$\lvert h_{k,m,i}\rvert^2 \sim (\frac{1}{d_{k,m}})^4$
Noise power	$\sigma^2 = 1 \times 10^{-13}\,\mathrm{W}$
MU's transmit, static, and CPU power	$p_k^{\mathrm{trans}} = 0.2\,\mathrm{W}, p_k^{\mathrm{static}} = 0.1\,\mathrm{W}$ $p_k^{\mathrm{calc}} = 1.5\,\mathrm{W}$
Local CPU frequency	$f_k^{\mathrm{local}} = 2\,\mathrm{GHz}$
Willingness to pay	$\alpha_k = 3.56 \cdot 10^{-5}$
Large Task	$s_{k,i} = 12\,\mathrm{Mbit}, c_{k,i} = 500$
Medium Task	$s_{k,i} = 1\,\mathrm{Mbit}, c_{k,i} = 1000$
Small Task	$s_{k,i} = 150\,\mathrm{kbit}, c_{k,i} = 2000$
Exploration parameter	$\epsilon_i = \frac{0.1}{1.01^i}$
Delay probability	$\lambda = 0.1$

V. NUMERICAL EVALUATION

A. Simulation setup

For the simulations, the parameters listed in Table I are considered, unless otherwise specified. We consider $K = 65$ MUs, and $M = 6$ APs. The MUs are randomly positioned in an area of $1\,\mathrm{km}^2$ and the APs are placed in an equidistant grid. Each of the MUs has one computation task arriving in each time slot belonging to one of the following task types: small, medium, or large. The size and complexity of each task type are specified in Table I. Each of the MUs has a battery level a_k which is uniformly distributed in the interval $(0, 1]$. The battery level a_k determines the willingness to pay, i.e., the lower the battery level, the more the MU will pay for energy consumption reduction. The willingness to pay is calculated by

$$\alpha_k(a_k) = 3.56 \cdot 10^{-5}(1 - a_k)\frac{\text{Monetary Units}}{\mathrm{J}}, \qquad (17)$$

which is the amount of monetary units that an MU is willing to pay for the energy consumption reduction by 1 J. In each time slot i, the available CPU frequency $F_{m,i}^{\max}$ at each AP is drawn from a normal distribution with mean 30 GHz and a standard deviation of 5 GHz. For each figure, 15 Monte-Carlo iterations over a time horizon of $I = 500$ were performed.

We use the following algorithms as benchmarks for our proposed CA-TO-MAB. Assuming full information $\mathcal{I}_k^{\mathrm{MU}}$ for each MU, we consider the following offline approaches:

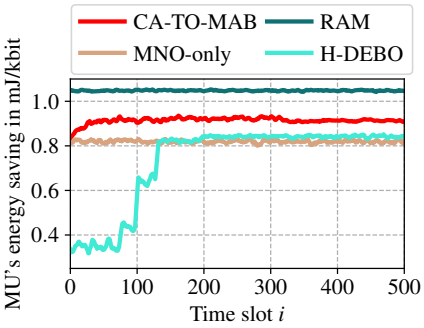

Fig. 2. Energy saved per kbit of all MUs.

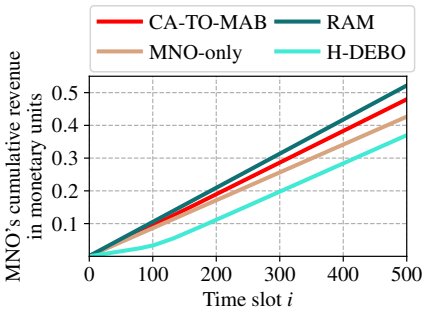

Fig. 3. Cumulative revenue of the MNO.

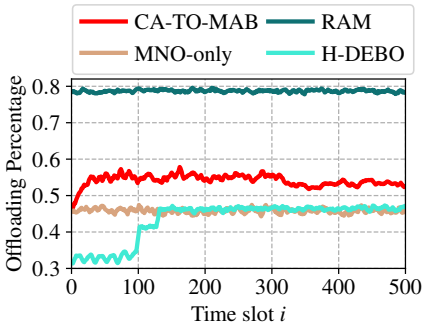

Fig. 4. Percentage of successfully offloaded tasks.

- *Resource Allocation Matching* (RAM): This algorithm is a modified version of the algorithm from [6]. This algorithm solves the task offloading game from Section III assuming complete information, i.e., MUs have complete information about the available resources. Each MU can calculate its preference ranking (12) in advance. In each time slot i, the MUs send offloading proposals to APs in the order of their preference ranking until an offloading proposal is accepted.
- *Only the MNO acts strategically* (MNO-only): Each MU proposes to a random AP in each time step. The MNO will select the MUs to maximize its revenue according to Algorithm 2. If the MU is rejected, it may send an offloading proposal to another random AP.

Additionally, we consider the following decentralized online learning benchmark, which does not require $\mathcal{I}_k^{\mathrm{MU}}$:

- *Heterogeneous Decentralized Epoch Based Offloading* (H-DEBO) [7]: The learning process of the MUs is split into three phases. In the exploration phase, the MUs are grouped and explore the resources in a round-robin fashion. In the second phase, the matching phase, MUs are matched to resources based on their experienced utility of (8) from the exploration phase. In the third phase, the exploitation phase, the MUs offload the task to the AP determined in the second phase. We modified the matching phase of the algorithm so that the time constraint (6) of each MU is fulfilled.

B. Results

Fig. 2 shows the average reduction of energy consumption of the MUs per $1\,\mathrm{kbit}$ of task size in each time slot. The RAM and MNO-only algorithms show constant performance over time, as both algorithms do not use online learning. Both algorithms require the full information already at the first time slot $i = 0$. The RAM algorithm shows the best performance with an average reduction of energy consumption of $1.03\,\mathrm{mJ}$ per kbit. The MNO-only algorithm achieves an energy reduction of $0.81\,\mathrm{mJ}$ per kbit. The H-DEBO algorithm has an exploration phase until $i = 100$, during which its performance is poor. In the exploration phase of H-DEBO, it is especially challenging to ensure that all MUs can fulfill the

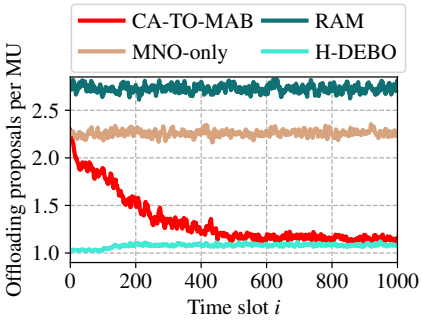

Fig. 5. Communication overhead caused by offloading proposals.

time constraint (6). Therefore, the number of offloading MUs is limited by the algorithm in the exploration phase at each AP to ensure that each MU can successfully offload its tasks. In the matching and exploitation phase, the performance significantly improves. The proposed CA-TO-MAB algorithm shows a performance comparable to the MNO-only algorithm in $i = 0$, as the MUs do not know their preference ranking (12) of APs. After convergence, the proposed CA-TO-MAB achieves 6% higher energy consumption reduction compared to H-DEBO.

Fig. 3 shows the cumulative revenue of the MNO in monetary units. The RAM and MNO-only algorithms show a linear performance over time, as both algorithms yield the same revenue of the MNO in each timestep. Both algorithms do not use online learning but require knowledge of $\mathcal{I}_k^{\mathrm{MU}}$ at each MU, therefore their revenue in each time slot is constant over time. The RAM algorithm shows the best performance with 0.52 monetary units after 500 time slots. The proposed CA-TO-MAB achieves the second highest MNO's cumulative revenue with 0.48 monetary units. The H-DEBO algorithm achieves a lower MNO's cumulative revenue of 0.38 monetary units, as it does not consider a techno-economic model. The H-DEBO algorithm is focused on the MU's utility function (8), therefore the MNO's cumulative revenue is $25\,\%$ lower at $i = 500$ than the MNO's cumulative revenue when using CA-TO-MAB. Thus, our proposed CA-TO-MAB algorithm increases both, the MUs' energy savings and the MNO's revenue, compared to the state-of-the-art H-DEBO algorithm.

Fig. 4 shows how many computation tasks are offloaded to

the MEC network. Using the RAM algorithm, 79 % of the MU's computation tasks are offloaded to the MEC network. The proposed CA-TO-MAB algorithm offloads 55 % of the MUs' tasks to the MEC network. The MNO-only and H-DEBO algorithm can offload 48% of the MU's tasks to the MEC network. The number of offloaded tasks shows how well the resources of the MEC network are utilized.

Fig. 5 shows the communication overhead required for each algorithm in terms of the number of offloading proposals per MU. The non-learning approaches require a significantly higher constant communication overhead. The RAM (MNO-only) algorithm requires on average 2.7 (2.2) offloading proposals per MU. These algorithms send multiple offloading proposals per MU, as the MUs do not learn to avoid collisions at the APs, and therefore it is required that MUs send offloading proposals to multiple APs. The proposed CA-TO-MAB algorithm starts with 2.2 offloading proposals per MU at $i = 0$, and then significantly decreases the number of offloading proposals. The MUs with a bad channel, i.e., low channel gain $|h_{k,m,i}|^2$, or low willingness α_k to pay learn that it is not beneficial to offload and consequently stop to send offloading proposals. Both online learning algorithms, CA-TO-MAB and H-DEBO, converge to 1.15 offloading proposals on average per MU, which significantly reduces the required communication overhead in the MEC network.

VI. Conclusion

In this work, we study decentralized online learning for computational task offloading in MEC. We consider two major challenges in MEC: (i) The techno-economic perspective of the MUs and the MNO and (ii) the limited information at the MUs about the resources and task offloading strategies of other MUs. To address the first challenge, we formulate a matching game that models the techno-economic interaction of the MNO and MUs. Furthermore, we consider the limited information of the MUs about the resources and the offloading strategies of other MUs. We propose a novel decentralized online learning algorithm for task offloading, termed CA-TO-MAB, which combines elements from online learning and game theory. In the numerical evaluation, we show that our proposed CA-TO-MAB algorithm decreases the energy consumption of MUs up to 10 % compared to the state-of-the-art H-DEBO online learning approach and, at the same time increases the cumulative revenue of the MNO by 25 %. Furthermore, the communication overhead can be reduced by 55% compared to the RAM algorithm. The extension to a dynamic system model with asynchronous task arrivals and the consideration of practical implementation aspects are left for future work.

References

[1] K. Jiang, H. Zhou, X. Chen, and H. Zhang, "Mobile Edge Computing for Ultra-Reliable and Low Latency Communications," *IEEE Communications Standards Magazine*, pp. 1–9, 2021.

[2] M. S. Z. Thu and E. C. Htoon, "Markov Based Computational Tasks Offloading Decision for Face Detection," in *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 2019, pp. 319–323.

[3] X. Xu, Q. Wu, L. Qi, W. Dou, S. B. Tsai, and M. Z. A. Bhuiyan, "Trust-Aware Service Offloading for Video Surveillance in Edge Computing Enabled Internet of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1787–1796, 2021.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[5] L. Zhang, J. Luo, L. Gao, and F.-C. Zheng, "Learning-Based Computation Offloading for Edge Networks with Heterogeneous Resources," in *IEEE Int. Conf. on Communications (ICC)*, 2020, pp. 1–6.

[6] B. Gu, Z. Zhou, S. Mumtaz, V. Frascolla, and A. K. Bashir, "Context-aware task offloading for multi-access edge computing: Matching with externalities," in *IEEE Global Communications Conf. (GLOBECOM)*, 2018, pp. 1–6.

[7] X. Wang, J. Ye, and J. C. Lui, "Decentralized Task Offloading in Edge Computing: A Multi-User Multi-Armed Bandit Approach," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 1199–1208.

[8] B. Yılmaz, A. Ortiz, and A. Klein, "Delay Minimization for Edge Computing with Dynamic Server Computing Capacity: A Learning Approach," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2020, pp. 1–6.

[9] A. U. Rahman, G. Ghatak, and A. De Domenico, "An online algorithm for computation offloading in non-stationary environments," *IEEE Communications Letters*, vol. 24, no. 10, pp. 2167–2171, 2020.

[10] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.

[11] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.

[12] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Game-theoretic learning-based qos satisfaction in autonomous mobile edge computing," in *2018 global information infrastructure and networking symposium (GIIS)*. IEEE, 2018, pp. 1–5.

[13] T. Mahn, D. Becker, H. Al-Shatri, and A. Klein, "A distributed algorithm for multi-stage computation offloading," in *IEEE 7th Int. Conf. on Cloud Networking (CloudNet)*, 2018, pp. 1–6.

[14] J. Du, L. Zhao, J. Feng, X. Chu, and F. R. Yu, "Economical revenue maximization in cache enhanced mobile edge computing," in *2018 IEEE Int. Conf. on Communications (ICC)*, 2018, pp. 1–6.

[15] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2016.

[16] K. Bando, "Many-to-one matching markets with externalities among firms," *Journal of Mathematical Economics*, vol. 48, no. 1, pp. 14–20, 2012.

[17] L. T. Liu, F. Ruan, H. Mania, and M. I. Jordan, "Bandit Learning in Decentralized Matching Markets," *J. Mach. Learn. Res.*, vol. 22, pp. 211–1, 2021.

[18] A. Krause and D. Golovin, "Submodular function maximization," *Tractability*, vol. 3, pp. 71–104, 2014.