Yi Wang, Anja Klein, and Lin Xiang, “Learning-Guided Matching Game for Decentralized Task Offloading to Multi-Functional UAVs” in *IEEE International Conference on Communications (ICC), Montreal, Canada, June 2025.*

Learning-Guided Matching Game for Decentralized Task Offloading to Multi-Functional UAVs

Yi Wang, Anja Klein, and Lin Xiang
Communications Engineering Lab, Technische Universität Darmstadt, Germany
Emails: {y.wang, a.klein, l.xiang}@nt.tu-darmstadt.de

***Abstract*—In this paper, we consider multiple multi-functional unmanned aerial vehicles (UAVs) with sensing, communication, and computation capabilities to simultaneously perform sensing and provide computing services to ground devices (GDs). The UAVs process both, the sensed data and the tasks offloaded by the GDs, onboard. To enable a scalable decentralized design, we formulate the task offloading problem as a matching game. In each time slot, each GD proposes to a UAV for task offloading, aiming to maximize its utility, defined as the time saved through offloading compared to local computing. Each UAV accepts to serve a subset of proposing GDs while handling its own sensing task as well. To balance resource allocation between offloaded and sensing tasks, each UAV aims to maximize its utility defined as a weighted sum of the total time saved for GDs minus the extra time incurred in processing its own sensing data when sharing partial computing resources with GDs compared to utilizing the entire computing resources for its sensing task. However, in practice, GDs may lack prior information about their channel conditions, the UAVs' communication and computing resources, and the task offloading strategies of other GDs, therefore, solving the task offloading matching game is challenging. To tackle this, we propose a novel gradient-based Multi-Armed-Bandit (GMAB) algorithm for task offloading, enabling GDs to learn and coordinate their offloading strategies in a decentralized manner. Simulation results show that the proposed GMAB algorithm outperforms several baseline task offloading schemes in terms of improving both, GD-side and UAV-side utility, by up to 34.1% and 37.2%, respectively.**

I. Introduction

Multi-functional unmanned aerial vehicles (UAVs) with joint sensing, communication and computation (JSCC) capabilities present a promising solution for on-demand, resource-efficient deployment and provisioning of sixth-generation (6G) wireless networks, applicable in both standard and emergency scenarios [1]. A fundamental research challenge, however, lies in efficiently scheduling interdependent sensing, communication, and computation tasks across these UAVs while managing inherently limited resources.

Several studies have explored centralized optimization for multi-functional UAVs, deploying them either as airborne edge servers (ESs) to provide computing services to ground devices [2], or as aerial access points performing JSCC [3], [4]. In [2], to minimize energy costs and latency incurred in offloading tasks from ground devices to the UAVs, transmit power, UAV selection, trajectory design, and onboard central processing unit (CPU) frequency allocation were jointly optimized. [3] optimized sensing scheduling, power allocation, and trajectories for UAVs performing integrated sensing and communication while offloading sensed data to a ground ES, aiming to reduce energy consumption and data collection time. However, both [2] and [3] assumed perfect knowledge of the global system characteristics such as wireless channel conditions, ES computation capabilities, and other devices' offloading decisions. These assumptions are often impractical in real-world applications. In contrast, [4] employed a deep reinforcement learning (RL) algorithm to jointly optimize a UAV's beamforming, computing resource allocation, and offloading ratio for maximization of computation throughput, where, unlike [3], the UAV could process part of the sensed data onboard. However, both [3] and [4] considered a single UAV, which is often limited in size, weight, and power. Deploying multiple (multi-functional) UAVs can mitigate these constraints, but the approaches [3], [4] become computationally intensive as the number of UAVs increases.

Decentralized optimization and learning provide a promising solution for *scalable* task scheduling across multiple multi-functional UAVs without requiring global information. Decentralized online learning solutions, such as those in [5]–[7], further eliminate the need for a priori knowledge of system characteristics by continuously learning it from observations. In [5], a multi-user multi-armed bandit (MAB) approach was introduced to maximize the number of accepted tasks by ESs under strict delay constraints, without prior knowledge of server processing speed and channel conditions. The authors of [6] applied a multiagent RL approach to optimize users' offloading decisions for minimizing the users' energy consumption and delay of task offloading. While [5], [6] focus on user-centric objectives such as delay and energy consumption, they do not consider ES-side objectives. In [7], a learning-guided matching approach [8] was employed to optimize decentralized decisions for task offloading and acceptance, while balancing users' goal of minimizing energy consumption and delay with ESs' aim for revenue maximization.

In this paper, we consider a scenario in which multi-functional UAVs are deployed to provide computing services for ground devices (GDs) while the UAVs simultaneously perform sensing tasks. Each UAV's onboard CPU manages the processing for both its own sensing tasks and the tasks offloaded from the GDs. Each GD selects a UAV for potential task offloading to minimize its task completion time, or equivalently, to maximize GD's utility, defined as the time saved through offloading compared to local computing. To balance resource allocation between these tasks, each UAV accepts a subset of proposing GDs to maximize its utility, defined as a weighted sum of the total time saved for GDs minus the extra time required for processing its own sensing data when sharing partial computing resources with GDs compared to utilizing the entire computing resources for its sensing task. To enable a scalable decentralized solution, we formulate the task offloading problem as a matching game. To solve this problem without a priori information at the GDs, we integrate learning techniques into the matching game, and propose a novel gradient-based multi-armed bandit algorithm (GMAB) for the GDs' task proposal. Our contributions are as follows:

This work has been funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center under grant LOEWE/1/12/519/03/05.001(0016)/72 and has been supported by the BMBF project Open6GHub under grant 16KISKO14.

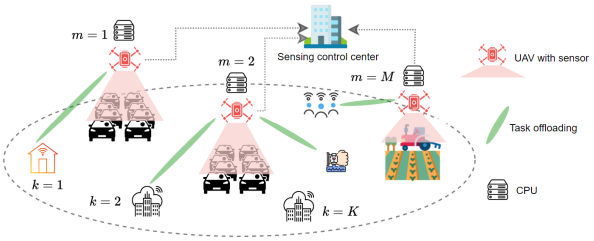


Fig. 1. Illustration of simultaneous onboard sensing and processing of sensed data and offloaded tasks using multiple multi-functional UAVs.

- We employ multiple multi-functional UAVs with sensing, communication and computation capabilities to perform sensing while providing computing services to GDs, and process both, sensed data and tasks offloaded by GDs, using the UAVs' onboard CPU. To enable a decentralized solution, we formulate the task offloading problem as a matching game between GDs and UAVs.
- To tackle the challenge of solving the problem of GDs lacking prior information about system characteristics, (e.g., channel condition, the UAV's communication and computing resources, and the task offloading strategies of other GDs), we propose a novel gradient-based multi-armed bandit algorithm, GMAB, which enables GDs to learn and coordinate their offloading strategies in a decentralized manner.
- Simulation results show that, compared with several baseline schemes, the proposed task offloading algorithm, GMAB, can simultaneously improve GD-side and UAV-side utilities by up to 34.1% and 37.2%, respectively.

II. System Model

This section presents the system model for decentralized task offloading from GDs to multi-functional UAVs. We further analyze the delays experienced in processing the GDs' tasks and the UAVs' sensed data.

A. Decentralized Task Offloading to Multi-Functional UAVs

As illustrated in Figure 1, we consider M multi-functional UAVs, denoted by UAV_m, $m \in \mathcal{M} \triangleq \{1, \cdots, M\}$, deployed by a sensing control center to perform sensing tasks, such as target detection and traffic estimation, in a given area while hovering over designated positions. Each UAV is equipped with a sensor and a CPU. The UAVs process the sensed data onboard before sending the results to the sensing control center. Additionally, the UAVs serve as ESs to compute tasks offloaded from K GDs scattered across the area of interest, which are denoted by GD_k, $k \in \mathcal{K} \triangleq \{1, \cdots, K\}$. The system operates in discrete time slots, indexed by $t = 1, \cdots, T$. During each time slot, each GD may generate a computation task that can either be processed locally by itself or offloaded to a UAV over the wireless channel.

At the beginning, the UAVs inform the GDs about their availability. The considered decentralized task offloading involves two phases, task proposal and task acceptance. In the task proposal phase, a GD initiates an offloading attempt by sending a small offloading proposal, including task information such as the GDs' task size, computing complexity, estimated local computing time, transmit power, along with pilots to its selected UAV. These pilots enable the selected UAVs to estimate the GDs' channels (however, we assume no feedback of channel estimates to the GDs). Each UAV may receive offloading proposals from multiple GDs. In the task acceptance phase, each UAV selects a subset of these proposals to accept and notifies the corresponding GDs of the acceptance status. The criteria and procedure for selecting the UAVs and the GDs in the two phases will be elaborated in the remainder of this section and in Sec. III. If a UAV accepts a GD's offloading proposal, the GD transmits the task data to the UAV; otherwise, the task will be computed locally by the GD.

Within each time slot, each UAV shares its computing resources between processing its own sensed data as well as the offloaded tasks from GDs, where the processed results are sent to the sensing control center and the corresponding GDs, respectively. Due to the significantly smaller size of the results compared to the original task data, the transmission time for the results is assumed to be negligible. Similarly, the delays in sending offloading proposals (alongside pilots) and UAVs' decisions can also be neglected.

We use binary variable $x_{k,m,t} \in \{0, 1\}$ to indicate whether GD_k offloads its task to UAV_m in time slot t. $x_{k,m,t} = 1$ indicates that UAV_m accepts the task offloading proposal of GD_k in time slot t. If GD_k either does not send a task offloading proposal to UAV_m or send one that is rejected, then, $x_{k,m,t} = 0$. The task offloading matrix $\boldsymbol{X}_t \triangleq \{x_{k,m,t}, k \in \mathcal{K}, m \in \{0, \mathcal{M}\}\}$ contains the information where each task is computed in time slot t.

B. GD's Utility

The task of GD_k in time slot t is characterized by its data size $D_{k,t}$ (bits) and computing complexity $c_{k,t}$ (CPU cycles per bit) [9]. The primary incentive for GDs to offload tasks is the potential reduction in task completion time. If the task is processed locally by GD_k, the required computing time is

$$\tau_{k,t}^{\mathrm{local}} = \frac{D_{k,t} \cdot c_{k,t}}{f_k^{\mathrm{GD}}}, \tag{1}$$

where f_k^{GD} denotes the CPU frequency of GD_k. Alternatively, if the task is offloaded from GD_k to UAV_m, the required communication and computation times are given by

$$\tau_{k,m,t}^{\mathrm{comm}} = \frac{D_{k,t}}{r_{k,m,t}}, \quad \text{and} \quad \tau_{k,m,t}^{\mathrm{cmp}} = \frac{D_{k,t} \cdot c_{k,t}}{f_{k,m,t}}, \tag{2}$$

respectively, where $r_{k,m,t}$ is the achievable data rate over the wireless channel from GD_k to UAV_m in time slot t, and $f_{k,m,t}$ denotes the CPU frequency allocated to GD_k by UAV_m in time slot t. Thus, by offloading the task to UAV_m, the time savings for GD_k can be expressed as

$$\tau_{k,m,t}^{\mathrm{saved}} = \tau_{k,t}^{\mathrm{local}} - \tau_{k,m,t}^{\mathrm{comm}} - \tau_{k,m,t}^{\mathrm{cmp}}. \tag{3}$$

If $\tau_{k,m,t}^{\mathrm{saved}} > 0$, offloading is beneficial; otherwise, when $\tau_{k,m,t}^{\mathrm{saved}} \leq 0$, GD_k opts for local computing.

The utility function of GD_k is defined as the time saved by offloading its task to a UAV, given by

$$U_{k,m,t}^{\mathrm{GD}} = \begin{cases} \tau_{k,t}^{\mathrm{local}} - \tau_{k,m,t}^{\mathrm{comm}} - \tau_{k,m,t}^{\mathrm{cmp}}, & m \in \mathcal{M}, \\ 0, & m = 0, \end{cases} \tag{4}$$

where $m = 0$ denotes local computing. With a slight abuse of notation, we also represent local computing as offloading to a virtual UAV denoted by UAV_0.

The channel gain for transmitting task data from GD_k to UAV_m in time slot t, denoted by $H_{k,m,t}$, is modeled as

$$H_{k,m,t} = \sqrt{A_0} \cdot (d_{k,m})^{-a/2} \cdot h_{k,m,t}, \quad (5)$$

where $d_{k,m}$, $A_0(d_{k,m})^{-\alpha}$, and $h_{k,m,t} \in \mathbb{C}$ denote the distance, path loss, and channel fading between GD_k and UAV_m, respectively. We consider a block fading channel model, where $h_{k,m,t}$ keeps constant within one time slot, but may change between time slots.

Each UAV is assigned a dedicated, non-overlapping frequency band with bandwidth $B_m^{\max}$, which is divided into orthogonal frequency sub-channels as in frequency-division multiple access (FDMA). These sub-channels are allocated to GDs that the UAV serves. Let $b_{k,m,t}$ be the bandwidth allocated to GD_k by UAV_m in time slot t, and $p_{k,t}$ be the transmit power of GD_k in time slot t. The achievable data rate $r_{k,m,t}$ is given as

$$r_{k,m,t} = b_{k,m,t} \cdot \log_2 \left(1 + \frac{|H_{k,m,t}|^2 \cdot p_{k,t}}{N_0 \cdot b_{k,m,t}}\right), \quad (6)$$

where N_0 is the power spectral density of the noise at UAVs.

C. UAV's Utility

Allocating a portion of computing resources to GDs reduces the resources available for the UAV's own sensing tasks. Let $D_{m,t}^{\mathrm{s}}$ (bits) and $c_{m,t}^{\mathrm{s}}$ (CPU cycles per bit) represent the size and computing complexity of the sensed data by UAV_m, respectively. Due to the task offloading of GDs, the extra time required for computing its sensing task is

$$\tau_{m,t}^{\mathrm{extra}} = D_{m,t}^{\mathrm{s}} c_{m,t}^{\mathrm{s}} \left(\frac{1}{f_{m,t}^{\mathrm{s}}} - \frac{1}{F_m^{\max}}\right), \quad (7)$$

where $f_{m,t}^{\mathrm{s}}$ is the CPU frequency allocated to the sensing task in time slot t. Given that $f_{m,t}^{\mathrm{s}} \le F_m^{\max}$, we have $\tau_{m,t}^{\mathrm{extra}} \ge 0$.

The UAVs aim to provide computing services to GDs while simultaneously processing their own sensing tasks using the onboard CPU. Let $\mathcal{A}_{m,t}$ denote the set of GDs whose tasks are accepted and processed by UAV_m in time slot t. Specifically, GD_k is in the set $\mathcal{A}_{m,t}$ if $x_{k,m,t} = 1$. To balance the computational resource allocation for GDs' tasks and the UAV's sensing task, we design the UAV's utility as a weighted sum of the total time savings for the GDs in set $\mathcal{A}_{m,t}$ minus the extra time incurred in processing sensing data, which is given by

$$U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t}) = w_m \sum_{k \in \mathcal{K}} x_{k,m,t} \cdot \tau_{k,m,t}^{\mathrm{saved}} - (1 - w_m)\tau_{m,t}^{\mathrm{extra}}, \quad (8)$$

where $0 \le w_m \le 1$ and $1 - w_m$ are the weights assigned to the two missions at UAV_m.

III. Matching Game based Problem Formulation

A. Matching Game Problem and Stable Matching Solution

To optimize the task offloading matrix $\boldsymbol{X}_t$ in a decentralized manner, we formulate a matching game played between GDs and UAVs, which make rational decisions on task offloading proposals and acceptance based on their own preferences, respectively. The resulting task offloading game, denoted as $\mathcal{G}_t$, is formally described by a tuple $\mathcal{G}_t = (\mathcal{K}, \mathcal{M}, \succeq_k^{\mathrm{GD}}, \succeq_m^{\mathrm{UAV}})$ containing the set $\mathcal{K}$ of GDs, the set $\mathcal{M}$ of UAVs, the GDs' preference $\succeq_k^{\mathrm{GD}}$ and the UAVs' preference $\succeq_m^{\mathrm{UAV}}$. The preference of GD_k is determined by its utility function, expressed as

$$m \succeq_k^{\mathrm{GD}} n \Longleftrightarrow U_{k,m,t}^{\mathrm{GD}} \ge U_{k,n,t}^{\mathrm{GD}}, \quad m, n \in \{0, \mathcal{M}\}. \quad (9)$$

In (9), GD_k prefers UAV_m over UAV_n if selecting UAV_m to compute its task yields a higher utility than selecting UAV_n.

Additionally, the preference of UAV_m is given by

$$\mathcal{A} \succeq_m^{\mathrm{UAV}} \mathcal{A}' \Longleftrightarrow U_{m,t}^{\mathrm{UAV}}(\mathcal{A}) \ge U_{m,t}^{\mathrm{UAV}}(\mathcal{A}'), \quad m \in \mathcal{M}, \quad (10)$$

Based on (10), UAV_m prefers the set of GDs in $\mathcal{A}$ over those in $\mathcal{A}'$ if UAV_m achieves higher utility by accepting the task proposals from GDs in $\mathcal{A}$ than in $\mathcal{A}'$, with the resource allocation $\boldsymbol{b}_{m,t} \triangleq \{b_{k,m,t} \ge 0, \forall k\}$ and $\boldsymbol{f}_{m,t} \triangleq \{f_{m,t}^{\mathrm{S}} \ge 0, f_{k,m,t} \ge 0, \forall k\}$ satisfying the following constraints:

(i) $\sum_{k \in \mathcal{K}} b_{k,m,t} \le B_m^{\max}$, ensuring that allocated bandwidth does not exceed the total available bandwidth of UAV_m.

(ii) $f_{m,t}^{\mathrm{S}} + \sum_{k \in \mathcal{K}} f_{k,m,t} \le F_m^{\max}$, ensuring the total allocated CPU frequency for UAV_m's sensing task and GDs' tasks accepted by UAV_m, remains within the UAV_m's available CPU capacity.

The solution to the defined task offloading game $\mathcal{G}_t$ is characterized by a *stable matching*, i.e., a task offloading matrix $\boldsymbol{X}_t$, in which no blocking pair exists. It is important to note that the stable matching may not be unique; indeed, multiple solutions may exist. A pair $(\mathrm{UAV}_m, \mathcal{A})$ is a blocking pair if UAV_m and GDs in set $\mathcal{A}$ can further enhance their individual utility by changing the current matching. Formally, the pair $(\mathrm{UAV}_m, \mathcal{A})$ is a blocking pair if the following three conditions are fulfilled:

(i) $\mathcal{A} \backslash \mathcal{A}_m \ne \emptyset$, i.e., not all GDs in $\mathcal{A}$ are currently matched to UAV_m.

(ii) $\mathcal{A} \succeq_m^{\mathrm{UAV}} \mathcal{A}_m$, i.e., UAV_m prefers to have GDs in set $\mathcal{A}$ matched over its current matched GDs in set $\mathcal{A}_m$.

(iii) $\mathrm{UAV}_m \succeq_k^{\mathrm{GD}} \mathrm{UAV}_n$, i.e., all GDs in $\mathcal{A}$ prefer to be matched to UAV_m over their current matched UAV.

B. Lack of Prior Information at GDs

For practical applicability, this paper assumes that GDs do not have prior information about the communication and computation resources available at each UAV, nor the channel conditions of their communication links with each UAV. Furthermore, GDs cannot predict the resources ultimately allocated to them, as these allocations depend on the offloading strategies of other GDs and the UAVs' preferences for different GDs. Given that GDs do not have such prior information, obtaining the stable solution requires further integrating learning into the game theoretical solution [8]. GDs have to estimate the time savings by interacting with UAVs, as defined by

$$\bar{U}_{k,m,t}^{\mathrm{GD}} = \mathbb{E}\{U_{k,m,t}^{\mathrm{GD}}\}, \quad m \in \mathcal{M}. \quad (11)$$

In Sec. III, we introduce a novel learning-based framework that enables each GD to independently learn its task offloading strategy by observing the rewards, specifically, the time saved by offloading its task to UAV_m, $m \in \{0, \mathcal{M}\}$. To this end, we consider the task offloading problem as a multi-user multi-armed bandit problem, where each of the K GDs has $M + 1$ arms, i.e., it can either offload its task to one of the M UAVs or choose local computing. However, due to variations in the task size, computing complexity of GDs' and UAVs' tasks, and fluctuations in channel capacity, the estimation of time savings is initially unreliable during the learning process. To address this challenge, we propose a GMAB-based task proposal algorithm.

Algorithm 1: GMAB (GD's strategy)

1 Initialize $J_{k,m,t=1} = 1$, $\forall k \in \mathcal{K}$, $\forall m \in \{0, \mathcal{M}\}$;
2 **for** $t = 1, \cdots, T$ **do**
3 $\mathcal{X} = \emptyset$;
4 **for** $k = 1, \cdots, K$ **do**
5 Select m based on $\pi_{k,m,t}$ in (12);
6 **if** $m \neq 0$ **then**
7 Send task offloading proposal to UAV_m and $\{D_{k,t}, c_{k,t}, \tau_{k,t}^{\mathrm{local}}, p_{k,t}\}$;
8 **else**
9 Process the task locally, and set $R_{k,t} = 0$;
10 $\mathcal{X} = \mathcal{X} \cup k$;
11 **end**
12 **end**
13 **for** $k = \mathcal{K} \backslash \mathcal{X}$ **do**
14 Wait for the decision $x_{k,m,t}$ of the selected UAV_m in Algorithm 2;
15 **if** $x_{k,m,t} = 1$ **then**
16 Transmit task of $D_{k,t}$ bits to UAV_m, observe $R_{k,t}$;
17 **else**
18 Process the task locally, and set $R_{k,t} = 0$;
19 **end**
20 **end**
21 Update $J_{k,m,t}$ and $\pi_{k,m,t}$, $\forall k \in \mathcal{K}, \forall m \in \{0, \mathcal{M}\}$;
22 **end**

IV. Proposed Learning-Based Framework

In this section, we introduce the learning-based framework, which consists of the GMAB based task proposal algorithm for GDs and task acceptance algorithm for UAVs, to obtain a stable solution for the matching game with no a priori information at GDs. This framework allows GDs to learn and coordinate their task offloading strategies in a decentralized manner. Specifically, we first present the GMAB algorithm for finding the optimal task proposal strategy for each GD. Then, we give a simple task acceptance strategy for the UAV. Our approach is motivated by [10], which demonstrated the plausibility of the learning-guided matching solution through repeated and dynamic interactions among players, guided by simple decision-making rules [8].

A. GMAB Algorithm for Task Proposal

We apply a *soft-max distribution* (i.e., Gibbs or Boltzmann distribution) to determine the probability of choosing UAV_m at GD_k in time slot t [11]

$$\pi_{k,m,t} = \frac{e^{J_{k,m,t}}}{\sum_{m=0}^{M} e^{J_{k,m,t}}}, \quad m \in \{0, \mathcal{M}\}, \tag{12}$$

where $J_{k,m,t}$ denotes the numerical preference of GD_k for proposing to UAV_m, and $J_{k,m,t}$ is determined based on observed rewards in previous time slots. Specifically, after UAV_m processes the task of GD_k and the reward $R_{k,t} = \tau_{k,m,t}^{\mathrm{saved}}$ (i.e., saved time) is observed, the numerical preferences of GD_k for $m \in \{0, \mathcal{M}\}$ in time slot t are updated by [11]

$$\begin{aligned} J_{k,m,t+1} &= J_{k,m,t} + \eta(R_{k,t} - \bar{R}_{k,t})(1 - \pi_{k,m,t}), \\ J_{k,n,t+1} &= J_{k,n,t} - \eta(R_{k,t} - \bar{R}_{k,t})\pi_{k,n,t}, \forall n \neq m, \end{aligned} \tag{13}$$

where $\eta \geq 0$ is a step-size parameter, $\bar{R}_{k,t}$ is the average reward up to time $t-1$ of GD_k, $\bar{R}_{k,t} = (\sum_{\tau=1}^{t-1} R_{k,\tau})/(t-1)$. By adjusting $\pi_{k,m,t}$, the GMAB also balances between exploring new arms and exploiting optimal arms. With more rewards observed, GDs gain greater certainty about their optimal choice (i.e., a higher probability of selecting the best UAV_m), gradually shifting the focus towards exploitation.

Algorithm 2: Task acceptance (UAV_m's strategy)

1 Initialize $\mathcal{A}_{m,t} = \{\}$, $W_{m,t} = \{\}$ ($W_{m,t}$ is a dictionary consisting of index of proposing GD_k (key) and its marginal contribution $U_{m,t}^{\mathrm{UAV}}(\{k\})$ (value));
2 **for** *each* $k \in \mathcal{K}_{m,t}$ **do**
3 $W_{m,t} \leftarrow W_{m,t} + \{(k, U_{m,t}^{\mathrm{UAV}}(k))\}$;
4 **end**
5 **for** *each* w *in* $W_{m,t}$ **do**
6 $W_{\max} \leftarrow$ maximum value of $W_{m,t}$
7 $k_{\max} \leftarrow$ key of $W_{\max}$;
8 Calculate $U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t} \cup \{k\})$ and $U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t})$;
9 **if** $U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t} \cup \{k\}) \geq U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t})$ **then**
10 $\mathcal{A}_{m,t} \leftarrow \mathcal{A}_{m,t} \cup \{k\}$;
11 **end**
12 $W \leftarrow W - \{(k, max)\}$;
13 **end**
14 Publish the decisions $\{x_{k,m,t}\}$ to the proposing GDs in set $\mathcal{K}_{m,t}$;

Algorithm 1 describes the task proposal algorithm, GMAB. First, the numerical preferences are initialized. In time slot t, each GD selects m based on the probability determined in (12) (cf. line 5). If GD_k selects $m \neq 0$, it sends a task offloading proposal to the selected UAV_m including the relevant information $\{D_{k,t}, c_{k,t}, \tau_{k,t}^{\mathrm{local}}, p_{k,t}\}$ (cf. lines 6-7). If $m = 0$, GD_k processes its task locally and set $R_{k,t} = 0$ (cf. line 9). For GDs that choose $m \neq 0$, they wait for the decision from the selected UAV, as determined by Algorithm 2 (cf. line 14). If GD_k's task offloading proposal is accepted, it transmits its task to the selected UAV and observes reward $R_{k,t}$ (cf. line 16). Otherwise, it processes the task locally and set $R_{k,t} = 0$ (cf. line 18). Based on the observed rewards, the numerical preferences $J_{k,m,t}$ and the softmax distribution $\pi_{k,m,t}$ of all GDs for $\{0, \mathcal{M}\}$ are updated according to (13) and (12), respectively (cf. line 21).

B. Task Acceptance and Resource Allocation

Upon receiving proposals from proposing GDs, denoted as $\mathcal{K}_{m,t}$, each UAV selects a subset $\mathcal{A}_{m,t} \subseteq \mathcal{K}_{m,t}$ of proposing GDs, and optimizes communication and computation resource allocation to each accepted GD subject to the limited resources for maximizing its utility function. The utility maximization problem for UAV_m in time slot t can be formulated as

$$\begin{aligned} \mathcal{P}_1 : & \max_{\mathcal{A}_{m,t}, \boldsymbol{b}_{m,t}, \boldsymbol{f}_{m,t}} U_{m,t}^{\mathrm{UAV}}(\mathcal{A}_{m,t}) \\ & \text{subject to: } \mathrm{C}_1 : \mathcal{A}_{m,t} \subseteq \mathcal{K}_{m,t}, \\ & \mathrm{C}_2 : x_{k,m,t} \in \{0,1\}, \forall k, \\ & \mathrm{C}_3 : \sum\nolimits_{k \in \mathcal{K}} b_{k,m,t} \leq B_m^{\max}, \\ & \mathrm{C}_4 : f_{m,t}^{\mathrm{S}} + \sum\nolimits_{k \in \mathcal{K}} f_{k,m,t} \leq F_m^{\max}, \end{aligned} \tag{14}$$

where constraint C_1 guarantees that UAV_m can only accept the task offloading proposals from the proposing GDs in set $\mathcal{K}_{m,t}$. C3 and C4 guarantee that the allocated bandwidth and computational resources to GDs do not exceed the total available bandwidth and computational capacity of the UAVs, respectively. Problem $\mathcal{P}_1$ is a mixed-integer nonlinear programming (MINLP) problem, which is difficult to be optimally solved within polynomial time.

To address this challenge, we apply a low-complexity algorithm based on greedy approximation [12] to solve problem $\mathcal{P}_1$. Algorithm 2 describes the task acceptance strategy of each UAV based on the greedy approximation algorithm. First, the marginal contribution of each proposing GD in $\mathcal{K}_{m,t}$,

TABLE I
PARAMETER SETTINGS

Parameter	Value
Available bandwidth at each UAV	$B_m^{\max} = 1$ MHz
Available CPU frequency at each UAV	$F_m^{\max} = 5$ GHz
GD's local CPU frequency	$f_{k,t}^{\mathrm{GD}} = 0.5$ GHz
GD's transmit power	$p_{k,t} = 0.2$ W
Path loss exponent	$\alpha = 2$
Noise power	$N_0 = 1 \times 10^{-13}$ W
Path loss at the reference distance	$A_0 = -20$ dB
Step size	$\eta = 2 \times 10^{-2}$

denoted as $U_{m,t}^{\mathrm{UAV}}(\{k\})$, is calculated, considering that only GD_k is accepted by UAV_m (cf. lines 2-4). Next, UAV_m ranks the proposing GDs based on their respective marginal contributions in a descending order. Following this ranking list (cf. lines 6-7), UAV_m accepts a GD if its utility increases upon accepting this GD, otherwise, the GD's task offloading proposal will be rejected (cf. lines 8-12). To further reduce the computing complexity, we assume that the GDs accepted by UAV_m and its sensing task share UAV_m's resources equally [7]

$$b_{k,m,t} = \frac{B_m^{\max}}{|\mathcal{A}_{m,t}|}, \quad f_{k,m,t}^{\mathrm{S}} = f_{k,m,t} = \frac{F_m^{\max}}{|\mathcal{A}_{m,t}| + 1}. \tag{15}$$

Then, the UAV_m sends its decisions $\{x_{k,m,t}\}$ and the allocated bandwidth to the corresponding GD_k (cf. line 14).

V. Simulation Results

In this section, we evaluate the performance of the proposed task proposing scheme GMAB through simulations. We consider a scenario with $K = 100$ GDs and $M = 10$ UAVs randomly distributed over a $300\mathrm{m}^2$ area, with the UAVs positioned at a fixed height of 120 m. The size and computing complexity of the tasks of GDs and UAVs both are modeled as truncated Gaussian random variables. For GDs, the task size has a mean of 1 Mbit and a variance of 0.1 Mbit^2, bounded within $[0.2, 2]$ Mbit, while computing complexity has a mean of 4000 cycles per bit and a variance of 400 cycles per bit^2, bounded within $[800, 8000]$ cycles per bit. For UAVs, the task size has a mean of 10 Mbit and a variance of 1 Mbit^2, bounded within $[2, 20]$ Mbit. The computing complexity has a mean of 4000 cycles per bit and a variance of 400 cycles per bit^2, and is bounded by $[800, 8000]$ cycles per bit. The channel fading $h_{k,m,t}$ follows a Rician fading model with $\kappa = 1$ [13]. Unless stated otherwise, we set the simulation parameters according to Table I.

For comparison, we also evaluate four benchmark algorithms for GDs' task proposing strategies, while the UAVs accept GDs' task offloading proposals using Algorithm 2:

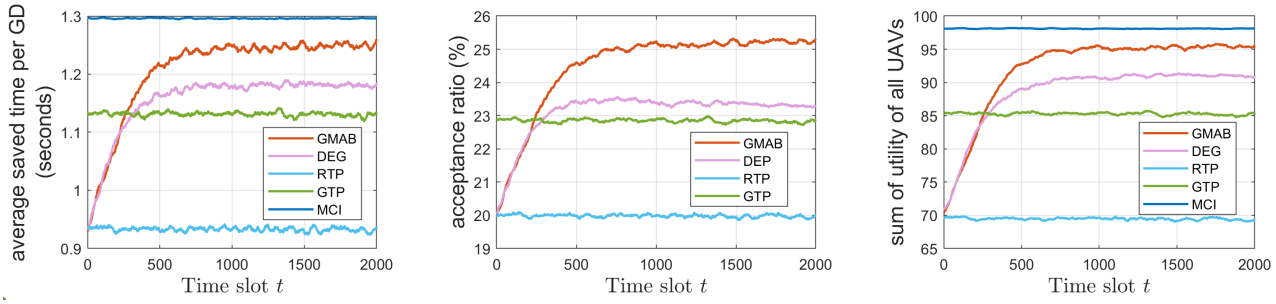
- *Matching with complete information (MCI)*: In each time slot t, each GD has complete information required to calculate the potential time savings achieved by offloading its task to each UAV based on (3). Each GD then generates a list of UAVs, ranked in a descending order of its preferences. The GD sends offloading proposals to the most preferred UAVs and stops if its proposal is accepted by one UAV or the list is exhausted. If the proposal is rejected, the GD proceeds with the next UAV in the list.
- *Random Task Proposal (RTP)*: Each GD proposes to a random UAV in each time slot.
- *Greedy Task Proposal (GTP)*: Each GD always proposes to the closest UAV. It is worth to note that the location information should not be available. This benchmark is intended to serve as a stronger reference for comparison.
- *Task offloading with decaying epsilon greedy (DEP) algorithm*: The GDs employ the decaying epsilon-greedy algorithm for task proposal, similar to [7]. In DEP, epsilon is initially set to a high value to prioritize exploration. Over time, epsilon gradually decreases, allowing the algorithm to shift its focus toward exploitation, leveraging the information gathered during the exploration phase.

Figure 2(a) shows the average saved time per GD of the considered schemes versus iterations over time. We observe that the average saved time of MCI, RTP and GTP is always constant over time, as they do not incorporate learning mechanisms. MCI outperforms all the other schemes due to the availability of complete information. GTP outperforms RTP because GDs propose to the closest UAV, which can reduce communication time when accepted. Through learning, GMAB and DEG gradually improve their performance, till ultimately outperforming RTP and GTP. This is because they allow GDs to strategically select UAVs based on the rewards observed from previous offloading decisions, enabling them to adapt and optimize their future offloading proposals. Specifically, GMAB achieves 6.3%, 10.7%, 34.1% higher average time savings per GD compared to DEG, GTP and RTP, respectively.

For insights into the performance gains of GMAB, Figure 2(b) illustrates the UAVs' acceptance ratio of task offloading proposals. We observe that the acceptance ratio of GMAB is higher than that of DEG. This is because, in GMAB, the numerical preference in (13) is updated even when a proposal is rejected by a UAV, since the average reward $\bar{R}_{k,t}$ changes. This allows GDs to prioritize UAVs based not only on average reward of each UAV but also on the likelihood of task proposal acceptance. In contrast, with DEP, GDs primarily propose to the UAV with the highest average reward, without considering the likelihood of proposal rejection by that UAV. If GDs consistently propose to UAVs with high rewards but low acceptance probabilities, this may lead to low long-term rewards, for which DEP is thus outperformed by GMAB.

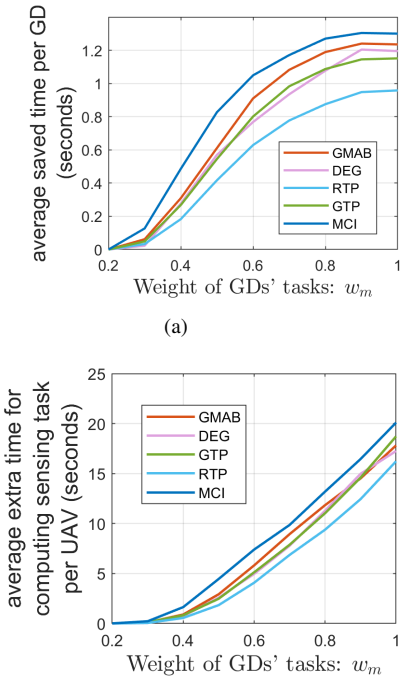
Figure 2(c) shows the sum of utility of all UAVs. With GMAB, the sum of utility of all UAVs increases with learning and closely approach that of MCI. This is because the UAVs accept task proposals based on their utility, which in turn optimize the task proposals from GDs in response to the UAVs' task acceptance decisions. As a result, the sum of utility of all UAVs increases over time. With GMAB, the sum of utility of all UAVs increases by 4.8%, 11.5%, and 37.2% compared to DEG, GTP, and RTP, respectively. This result highlights that the proposed learning-guided matching game framework facilitates mutual benefits for both parties.

Figure 3(a) further evaluates the average saved time per GD for different weights w_m assigned to GDs' tasks, where each UAV has the same w_m. As w_m increases, UAVs increasingly prioritize providing computing services to GDs in (8). Consequently, the average saved time per GD also increases. Regardless of the value of w_m, GMAB consistently outperforms all schemes except MCI which assumes perfect system knowledge. Notably, DEG performs poorly for small values of w_m, as its performance is close to GTP. However, when w_m is larger than 0.9, DEG surpasses GTP. For small



(a) (b) (c)

Fig. 2. (a) The average saved time per GD (b) The acceptance ratio (c) The sum of utilities of all UAVs.



(a)

(b)

Fig. 3. (a) Average saved time per GD. (b) Average extra time for computing sensing task per UAV.

values of w_m, DEG performs poorly because it does not allow GDs to adjust their task offloading strategies when a task is rejected. With small w_m, UAVs prioritize their own sensing tasks, leading to a reduced number of accepted offloading tasks, which ultimately reduces the performance of DEG.

Finally, Figure 3(b) shows the average extra time for computing the sensing task per UAV with respect to w_m. With a larger w_m, UAVs prioritize the GDs' offloaded tasks. Therefore, less computation resources are allocated to the UAVs' sensing tasks. Consequently, the extra time needed for computing the UAVs' sensing tasks increases with w_m. Notably, the performance of GMAB is very close to the performance of DEG and GTP for different values of w_m. This shows that the proposed GMAB algorithm can significantly reduce the task completion time of GDs, while having a comparable cost in the extra computation time of their own sensing tasks compared to other schemes.

VI. Conclusion

In this paper, we investigated decentralized task offloading from GDs to multi-functional UAVs, eliminating the need for GDs to have prior information of system characteristics, such as channel conditions, UAV's communication and computing resources, and the task offloading strategies of other GDs. To this end, we first formulated the task offloading problem as a matching game. In this framework, each GD's utility is defined by the time saved through offloading, while each UAV's utility is defined by a weighted sum of the total time saved for GDs minus the extra time spent on processing its own sensing data to balance resource allocation between GD tasks and UAV sensing tasks. To further address the challenge of no prior information at the GDs, we proposed a learning-based framework, consisting of a task proposal algorithm, GMAB, and a task acceptance algorithm, to obtain a stable matching for the formulated matching game. Simulation results showed that the proposed task offloading algorithm, GMAB, significantly outperforms several baseline schemes in simultaneously improving GDs' and UAVs' utilities by up to 34.1% and 37.2%, respectively.

References

[1] C. D. Alwis, et al., "Survey on 6G Frontiers: Trends, Applications, Requirements, Technologies and Future Research," *IEEE OJ-COMS*, vol. 2, pp. 836-886, 2021.

[2] F. Pervez, A. Sultana, et al., "Energy and Latency Efficient Joint Communication and Computation Optimization in a Multi-UAV-Assisted MEC Network," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 3, pp. 1728-1741, 2024.

[3] N. Huang, C. Dou, et al., "Unmanned-Aerial-Vehicle-Aided Integrated Sensing and Computation With Mobile-Edge Computing," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16830-16844, 2023.

[4] P. Qin, Y. Fu, et al., "URLLC-Aware Trajectory Plan and Beamforming Design for NOMA-Aided UAV Integrated Sensing, Communication, and Computation Networks," in *Pro. IEEE VTC*, 2024.

[5] X. Wang, et al., "Decentralized Task Offloading in Edge Computing: A Multi-User Multi-Armed Bandit Approach," in *Proc. IEEE INFOCOM*, 2022.

[6] Z. Gao, L. Yang and Y. Dai, "Fast Adaptive Task Offloading and Resource Allocation in Large-Scale MEC Systems via Multiagent Graph Reinforcement Learning," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 758-776, 2024.

[7] B. Simon, H. Mehler and A. Klein, "Online Learning in Matching Games for Task Offloading in Multi-Access Edge Computing," in *Proc. IEEE ICC*, 2023.

[8] Z. Han, D. Niyato D, W. Saad, et al. "Game theory for Next Generation Wireless and Communication Networks: Modeling, Analysis, and Design[M]," *Cambridge University Press*, 2019.

[9] T. Mahn, D. Becker, H. Al-Shatri, and A. Klein, "A Distributed Algorithm for Multi-Stage Computation Offloading," in *Proc. IEEE 7th CloudNet*, 2018.

[10] D. G. Harper, "Competitive Foraging in Mallards: Ideal Free Ducks," *Animal Behaviour*, vol. 30, no.2, pp. 575-584, 1982.

[11] R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction[M]," *MIT Press*, 2018.

[12] A. Krause and D. Golovin, "Submodular Function Maximization," *Tractability*, vol. 3, pp. 71–104, 2014.

[13] Y. Wang, M. Krantzik, L. Xiang and l. A. Klein, "Joint Communication and Computing Optimization for Digital Twin Synchronization with Aerial Relay," in *Proc. IEEE ICC*, 2024.